PARTS OF THE ORIGINAL DOCUMENT

ARE OF POOR REPRODUCTIVE QUALITY

CERTAINES PARTIES DU DOCUMENT ORIGINAL

SONT DE PIÈTRE QUALITÉ DE REPRODUCTION

UNIVERSITÉ DE MONTRÉAL

CENTRE DE RECHERCHE SUR LES TRANSPORTS

MÉMOIRE D'ÉTUDIANT

CASE POSTALE 6128, SUCCURSALE A, MONTRÉAL, QUÉBEC, H3C 3J7

CONTRIBUTIONS À LA THÉORIE ET AUX OUTILS DE PLANIFICATION DES RÉSEAUX DE TRANSPORT URBAIN

par

Heinz Spiess

Thèse de Ph.D., département d'informatique et de recherche opérationnelle, Université de Montréal, dirigée par Michael Florian.

Université de Montréal
Centre de recherche sur les transports - Publication #382
Octobre 1984

Dépôt légal - 1er trimestre 1985

TABLE DES MATIERES

Chapitre		Page
	LISTE DES FIGURES	v
	LISTE DES TABLEAUX	vii
	SOMMAIRE	viii
I.	INTRODUCTION	1
II.	CONTRIBUTIONS THEORIQUES	6
	1. La stratégie optimale pour le choix de route dans un réseau de transport en commun	6
	1.1 Introduction	6
	1.2 La stratégie de choix de route optimale	9
	1.3 Le modèle	20
	1.4 Le cas spécial des fréquences combinées	29
	1.5 Un algorithme de résolution	36
	1.6 Une extension non-linéaire	47
	1.7 Conclusions	54
	2. Un modèle de vraisemblance maximale pour l'estimation de matrices origine-destination	58
	2.1 Introduction	58
	2.2 Un modèle de vraisemblance maximale	62
	2.3 Un algorithme de résolution	68
	2.4 Le cas spécial des totaux marginaux connus	73
	2.5 L'inférence statistique	81
	2.6 L'extension du modèle aux volumes observés poissoniens	89
	2.7 Conclusions	98

III. EMME/2: CONCEPTION ET REALISATION D'UN LOGICIEL INTERACTIF-GRAPHIQUE POUR LA PLANIFICATION DE TRANSPORT URBAIN DE PERSONNES

1. Introduction 100

2. Concepts de base 105

2.1 L'approche interactive-graphique vs le traitement par lots 105
2.2 La structure du logiciel 111
2.3 Les entrées / sorties 115
2.4 La banque de données 125
2.5 L'interface entre l'usager et le système 138

3. Structures de données 149

3.1 Le réseau 151
3.2 Les matrices 164
3.3 Les fonctions 169
3.3 Données auxiliaires 175

4. Aspects graphiques 177

4.1 Le choix d'un système graphique de base 177
4.2 QPLOT - logiciel graphique de base 182
4.3 Le dessin 192
4.4 Les commandes graphiques 198
4.5 La feuille de travail graphique 201
4.6 Le rôle des couleurs 208

5. Les modules 212

5.1 Les utilitaires 214
5.2 L'éditeur de réseau 218
5.3 L'éditeur de matrices 231
5.4 L'éditeur de fonctions 238
5.5 Les modules d'affectation 242
5.6 L'analyse des résultats 261

6. Les applications 274

7. Conclusions 285

BIBLIOGRAPHIE 289

REMERCIEMENTS 297

LISTE DES FIGURES

Figure:		Page:
II-1-1	Exemple d'un réseau de transport en commun	15
II-1-2	La stratégie optimale	17
II-1-3	Le chemin optimal	18
II-1-4	La séquence optimale de noeuds de transfert	19
II-1-5	Représentation "noeuds-liens" du réseau de transport en commun	22
II-1-6	Réseau simplifié	23
II-1-7	Réseau avec temps et fréquences sur les liens	43
II-2-1	Distribution empirique de la statistique Y pour le problème d'estimation de matrice O-D avec volumes de liens connus, après 1000 essais Monte-Carlo	86
II-2-2	Distribution empirique de la statistique Y pour le problème d'estimation de matrice O-D avec totaux marginaux connus, après 1000 essais Monte-Carlo	87
II-2-3	Distribution de la statistique Y pour de petites valeurs de productions et d'attractions.	88
II-2-4	Distribution empirique de la statistique Y pour le problème d'estimation de matrice O-D avec volumes observés des liens, après 1000 essais Monte-Carlo	92
II-2-5	Distribution empirique de la statistique Y pour le problème d'estimation de matrice O-D avec totaux marginaux observés, après 1000 essais Monte-Carlo	96
III-2-1	Organisation verticale vs. horizontale	130
III-3-1	Contenu de la banque de données de EMME/2	149
III-3-2	Les éléments de réseau et leurs attributs	154
III-3-3	La structure de données associée aux virages	160
III-3-4	Les quatre formats de matrice	165

III-4-1	Représentation proportionnelle des volumes-autos sur le réseau régional	183
III-4-2	Agrandissement "ordinaire" du centre-ville	184
III-4-3	Agrandissement par changement de l'échelle géographique seulement	185
III-4-4	Format général d'un dessin	194
III-4-5	Format général d'une feuille de travail	204
III-4-6	Feuille de travail après une suite d'opérations	207
III-4-7	Feuille de travail regénérée	209
III-5-1	Entrée interactive-graphique des démarcations	217
III-5-2	Modification interactive-graphique du réseau de base	220
III-5-3	Réseau de base - modes permis sur les liens	221
III-5-4	Réseau de base - nb. de voies et fonction volume-délai	222
III-5-5	Le plus court chemin vers l'hôpital le plus proche	224
III-5-6	Le plus court chemin de l'université à l'aéroport	225
III-5-7	Modification interactive - graphique des lignes	227
III-5-8	L'affichage des itinéraires des lignes	229
III-5-9	Intersections et virages pénalisés	230
III-5-10	Déplacements vers le centre ville	233
III-5-11	Distribution des temps de déplacement	235
III-5-12	Fonctions volume - délai	240
III-5-13	L'analyse graphique d'une fonction de l'usager	241
III-5-14	Volumes - auto au centre-ville	263
III-5-15	Comparaison des volumes de deux scénarios	265
III-5-16	Volumes sur les virages des carrefours	267
III-5-17	Plus courts chemins et accessibilité des centroïdes	269
III-5-18	Détails des chemins du mode auto	271

III-5-19	Les volumes sur une ligne de transport en commun	272
III-5-20	Nombre d'embarquements initiaux aux noeuds	273
III-6-1	Nombre d'embarquements sur le réseau t.c. à Motréal	276
III-6-2	Fenêtre du réseau de base de la ville de Vancouver	277
III-6-3	Volumes sur les lignes de métro à Stockholm (Suède)	279
III-6-4	Le réseau de base de la ville de Portland (Orégon)	280
III-6-5	Itinéraire et volumes d'une ligne d'autobus à Trujillo (Pérou) avant l'étude de rationalisation	282
III-6-6	Volumes - auto dans la région métropolitaine de Paris	283

LISTE DES TABLEAUX

Tableau:		Page:
II-1-1	Tous les sous-ensembles de lignes attrayantes	16
II-1-2	Réseau exemple: Trouver la stratégie optimale	44
II-1-3	Réseau exemple: Affectation de la demande	45
II-1-4	Complexité des étapes de l'algorithme	45
II-2-1	Estimation d'une matrice avec totaux marginaux connus - un petit exemple	80
II-2-2	Variables duales et valeurs de la fonction objective	80
II-2-3	Solution optimale	81
III-2-1	Traitement par lot vs. traitement interactif - graphique	108
III-3-1	La table des noeuds	156
III-3-2	La conversation automatique entre les différents formats de matrice	167
III-3-3	Les priorités gauches et droites des opérateurs	174
III-5-1	Les modules de EMME/2	213

SOMMAIRE

Le sujet général de cette thèse est le problème de planification de réseaux de transport urbain des personnes.

Comme contributions théoriques, nous décrivons deux nouveaux modèles mathématiques: un modèle d'affectation de transport en commun et un modèle de vraisemblance maximale pour l'estimation d'une matrice origine - destination (O-D). Dans la deuxième partie de la thèse, nous décrivons la réalisation du système EMME/2, un outil interactif - graphique pour la planification de réseaux de transport urbain.

Les méthodes existantes pour l'affectation du transport en commun sont basées sur des méthodes heuristiques ou des modifications *ad hoc* de méthodes d'affectation pour le transport routier. Il n'existe pas de modèle mathématique satisfaisant pour ce problème. L'idée pricipale, qui distingue l'approche proposée des approches existantes, est basée sur l'observation que l'ensemble de choix possibles, auquel se voit confronté un passager du transport en commun, ne consiste pas simplement de l'ensemble de tous les chemins possibles, mais contient aussi des éléments composés. Nous dénotons un élément de l'ensemble de choix par le terme *stratégie*. Le problème d'affectation de transport en commun est alors formulé comme le problème de trouver la stratégie optimale, i.e. celle qui a un coût minimal. Nous montrons que, pour le cas spécial des fréquences combinées, le modèle peut être réduit à un problème de programmation linéaire, très similaire à un problème de chemins plus courts. Nous proposons un algorithme de résolution efficace de complexité polynomiale. Une extension du modèle, qui admet des coûts qui dépendent des volumes, est formulé. Nous montrons que ce modèle peut être reformulé comme un problème de programmation convexe et nous y adaptons un algorithme de résolution basé sur la méthode d'approximations linéaires successives.

La connaissance précise de la demande est un facteur crucial pour l'application de tout modèle de planification de réseau. A cause de contraintes budgetaires, souvent une enquête O-D est limitée à un très petit échantillon. Des données agrégées, comme la production et l'attraction des zones et les volumes de trafic sur les liens du réseau, sont souvent connues plus précisément, puisqu'en général, elles sont enquêtées soit exhaustivement, soit périodiquement. Nous proposons un modèle de vraisemblance maximale pour estimer une matrice O-D, à partir d'une matrice d'échantillon et des données agrégées sur le réseau. Nous supposons un processus d'échantillonnage qui donne lieu à des observations Poissonniennes. Les propriétés du modèle sont analysées et un algorithme est donné pour résoudre le modèle. A l'aide de simulations extensives de type Monte-Carlo, nous montrons que la distribution de la statistique Y peut être très bien approximée par une distribution χ^2. La statistique Y peut donc servir à tester la cohérence du modèle et nous donne un moyen quantitatif pour analyser la région de confiance de la matrice estimée.

Le système EMME/2 est un logiciel interactif-graphique pour la planification de réseaux de transport. Il comprend, d'une façon intégrée, un ensemble complet d'outils, pour la gestion, la modélisation et l'analyse de réseaux de transport multimodaux. Le but principal du développement de EMME/2 est de rendre plus facile l'utilisation de modèles de prévision complexes, en combinant des modèles théoriquement cohérents avec une implantation qui utilise la technologie informatique de pointe. Avec un tel système interactif - graphique, il est possible de libérer le planificateur des tâches fastidieuses reliées à l'exécution de programmes par lots, pour qu'il ait plus de temps pour s'occuper de la planification proprement dite. Dans ce nouveau environnement de travail, il est possible d'augmenter considérablement le nombre de scénarios qui peuvent être évalués dans un temps raisonnable. Nous discutons l'approche et les concepts de base utilisés dans le design du système EMME/2. En particulier, nous élaborons la conception de la banque de données et l'implantation du système graphique. Dans un aperçu général, nous décrivons brièvement le fonctionnement de chaque module, en illustrant les modules graphiques par des exemples de sortie provenant du réseau de la ville de Winnipeg. Finalement, nous présentons les diverses applications du système EMME/2 jusqu'à date.

I. INTRODUCTION

Dans les dix dernières années, le domaine des modèles de planification des réseaux de transport a été sujet à des développements théoriques fondamentaux, qui nous ont amené à une compréhension beaucoup plus profonde des phénomènes qu'on désire modéliser. On n'a qu'à penser à toutes les recherches reliées au problème d'affectation d'équilibre sur un réseau de transport.

Pendant la même période de temps, l'équipement informatique nécessaire pour pouvoir appliquer ces modèles de transport a subi des changements encore plus radicaux. Si, il y a dix ans, seulement les grands organismes avaient accès à un ordinateur, aujourd'hui des micro-ordinateurs d'une puissance de calcul comparable sont disponibles à un coût cent fois moins élevé, donc abordables même pour des agences de planification de transport de villes de petite et moyenne taille.

Ce progrès technologique permettrait d'intégrer beaucoup plus vite toutes les innovations théoriques dans l'environnement des praticiens dans les agences de planification de transport, et donc de diminuer l'écart existant entre l'état des connaissances théoriques et les méthodes appliquées en pratique. Malheureusement, on s'aperçoit que ceci n'est pas le cas. Au contraire, il semble que, pendant les dix dernières années, l'écart entre les théoriciens et les praticiens s'est nettement élargi.

Le travail présenté ici constitue une petite contribution à diminuer

cet écart. Deux modèles théoriques sont développés dans le chapitre II. Les deux correspondent à des besoins pratiques des planificateurs mais, en même temps, constituent des sujets qui sont théoriquement pertinents et intéressantes. Dans le chapitre III, nous décrirons la réalisation de EMME/2, un système interactif-graphique pour la planification de réseaux de transport urbain. Ce système est conçu comme un ensemble d'outils intégrés qui, sur le plan théorique, correspondent à l'état des connaissances actuelles, mais, sur le plan pratique, sont adaptés aux besoins d'un environnement réel et sont faciles à utiliser par le planificateur.

Dans la section 1 du chapitre II, nous présenterons une nouvelle approche au problème d'affectation sur un réseau de transport en commun. Récemment, ce problème n'a pas été le sujet de beaucoup de recherches, bien que toutes les méthodes utilisées en pratique sont basées sur des approches heuristiques et sont donc, au moins d'un point de vue théorique, insatisfaisantes. L'idée principale, qui distingue notre approche des approches existantes, est basée sur le fait que l'ensemble de choix possibles, auquel se voit confronté un passager du tranport en commun, ne consiste pas simplement de tous les différents chemins possibles, comme il est implicitement supposé dans les méthodes existantes. Puisque le temps d'attente diminue, si le passager est prêt à monter à bord du premier véhicule de plusieurs lignes, son ensemble de choix est considérablement plus grand que le nombre de chemins; en fait, il comprend tous les sous-ensembles de chemins possibles. Ceci nous amènera à la définition du terme *stratégie*, qu'on utilisera pour dénoter un élément de l'ensemble des choix possibles. Nous montrerons que le problème de calculer la stratégie optimale peut être formulé comme un problème d'optimisation qui, sous certaines conditions, se réduit à un problème de programmation linéaire, similaire à la formulation

du problème de plus courts chemins. Nous proposerons un algorithme de resolution efficace qui est de compléxité polynomial. On étudiera aussi l'extension de ce modèle pour le cas où le temps de voyage est une fonction croissante du volume de passagers transportés, et nous discuterons comment cette extension peut servir pour modéliser des effets de congestion. L'algorithme proposé a été implanté comme un module du système EMME/2 (voir plus loin); il est présentement utilisé par des agences de planification de transport dans les villes de Montréal, Vancouver, Stockholm et Portland (Orégon).

La connaissance précise de la demande est un facteur crucial pour le succès de tout modèle de planification de réseau. Malheureusement, la récolte de ce type de données coûte très cher. A cause de contraintes budgetaires, les enquêtes ne portent souvent que sur un très petit échantillon de voyageurs. Des données agrégées, comme la production et l'attraction des zones ou les volumes sur les liens du réseau, sont souvent connues plus précisément puisqu'en général, elles sont basées sur des enquêtes périodiques et / ou exhaustives. Des méthodes qui, à l'aide de cette information, permettent d'améliorer nos connaissances sur la matrice de demande sont donc d'un grand avantage pour le planificateur. Les méthodes existantes, la plupart basées sur l'approche de maximisation d'entropie, visent plutôt la synthèse d'une matrice de demande uniquement à partir des volumes sur le réseau, en l'absence d'une matrice échantillonnée. Bien que toutes ces méthodes permettent d'inclure de l'information a priori sur la matrice de demande, sous forme d'une "matrice de référence", elles font cela sans spécifier précisément la provenance de cette information. Ceci peut constituer un avantage dans certains cas, puisque cela permet l'utilisation de n'importe quel type d'information.

Par contre, cette approche rend impossible de faire des inférences statistiques sur la région de confiance de l'estimation résultante, ni sur la cohérence du modèle et des données. Si, par contre, on possède une matrice échantillonnée de provenance statistique connue, on peut y appliquer la méthode de vraisemblance maximale, une méthode bien connue en statistique mais qui, jusqu'à présent, n'a pas été appliquée au problème d'estimation de matrice. Dans la section 2 du chapitre II, nous poursuivrons cette idée et nous étudierons un modèle de vraisemblance maximale pour l'estimation d'une matrice origine - destination à partir des données agrégées sur le réseau. Nous supposerons, comme cas le plus important, que les éléments de la matrice observée correspondent à des observations de variables aléatoires qui suivent des distributions de Poisson. Nous analyserons les propriétés du modèle résultant et nous développerons un algorithme de résolution efficace. A l'aide de simulations extensives de type Monte-Carlo, nous montrerons que la statistique Y suit, à toute fin pratique, une distribution χ^2. Cette statistique peut donc servir pour tester la cohérence du modèle, et nous donne un moyen quantitatif pour analyser la région de confiance de la matrice résultante.

Dans le chapitre III, nous présenterons le système EMME/2, un logiciel interactif-graphique pour la planification de réseau de transport. La motivation principale pour ce projet est basée sur notre conviction ferme que, pour que des innovations théoriques puissent s'établir dans la pratique, il n'est pas suffissant de rester au niveau des théorèmes et des algorithmes, mais qu'il faut rendre ces résultats théoriques accessibles au praticien, sous forme d'*outils* qui sont faciles à utiliser dans un contexte de planification réelle. Il n'est pas du tout question d'automatiser le processus de planification; au contraire, le planificateur devrait avoir le

plus de contrôle possible, pour que le processus de planification puisse profiter au maximum de son expertise. L'énorme quantité de données nécessaires, ainsi que leur caractère spatial, forment un base idéale pour une approche interactive-graphique. En libérant le planificateur des tâches fastidieuses reliées à l'exécution de programmes de traitement par lots, il lui reste plus de temps pour faire ce qu'il fait le mieux: la planification proprement dite. Nous discuterons l'approche et les concepts de base utilisés dans le design du système EMME/2. Comme on le verra, le mode de travail interactif - graphique a des implications profondes sur tous les aspects du système. En particulier, nous élaborerons la structure modulaire du logiciel, la conception et le contenu de la banque de données, ainsi que les détails de l'implantation du système graphique. Sous forme d'un aperçu général, nous décrirons brièvement le fonctionnement de chaque module, en illustrant les modules graphiques par des exemples de sortie provenant du réseau de la ville de Winnipeg. Finalement, nous présenterons les diverses applications du système EMME/2 jusqu'à date. Bien que le développement est encore en cours, le système est déja installé dans les agences de planification de transport des villes de Montréal, Vancouver, Stockholm et Portland, et il a aussi servi dans le cadre de plusieurs projets de recherche au CRT.

II. CONTRIBUTIONS THEORIQUES

1. La stratégie optimale pour le choix de route dans un réseau de transport en commun

1.1 Introduction

Dans le présent chapitre, nous proposons un nouveau modèle, ainsi qu'un algorithme de résolution efficace, pour le problème de choix de route optimal dans un réseau de transport en commun avec des lignes à itinéraire fixe. Dans la première partie, nous étudierons le cas pour lequel le choix de route du voyageur n'est pas influencé par des effets de congestion. Tandis que le problème correspondant pour l'affectation des voitures privées est équivalent au problème bien connu de l'affectation "tout-ou-rien" sur le chemin le plus court, le problème devient beaucoup plus complexe dans le cas du transport en commun, cela étant dû au phénomène d'attente à l'arrivée d'un véhicule. Le temps d'attente est un élément important qui influence le choix de route d'un voyageur. Contrairement aux autres composantes d'un voyage dans un réseau de transport en commun (accès à l'arrêt, montée à bord d'un véhicule, séjour dans le véhicule, descente du véhicule, marche à la destination finale), le temps (ou coût) d'attente ne peut pas être modélisé comme un attribut additif associé avec l'activité d'attente de l'arrivée d'un véhicule d'une ligne. Cela devient évident quand on regarde le cas où un voyageur a décidé de prendre le premier

véhicule qui arrive parmi un ensemble de lignes; dans ce cas, l'espérance du temps d'attente dépend des caractéristiques de service sur les lignes considérées d'une façon non-additive. Puisqu'en plus, l'attente se fait *avant* de monter à bord d'un véhicule, elle constitue un élément *asymétrique* du voyage. La symétrie naturelle entre origine et destination qui existe dans le problème de trouver les chemins plus courts, et par conséquent aussi dans le problème d'affectation du transport routier, n'est plus présente dans le problème d'affectation du transport en commun.

Dans le passé, plusieurs auteurs ont étudié le problème d'affectation du transport en commun: les uns l'ont traité en soi (Dial, 1967; Chriqui, 1974; Rapp et al., 1976; Andreasson, 1976); d'autres l'ont abordé comme un sous-problème d'un modèle plus complexe, soit le problème de "design" d'un réseau de transport en commun (Lampkin et Saalmans, 1967; Scheele, 1977; Mandl, 1979; Hasselström, 1981), soit le problème d'affectation d'équilibre multimodal (Florian, 1977; Florian and Spiess, 1981). Basé sur ces études, des logiciels ont été développés, comme par exemple IGTDS (General Motors, 1978), TRANSCOM (Chapleau, 1974), TRANSEPT (Last and Leak, 1976), NOPTS (Rapp et al., 1976), le système VOLVO (Andreasson, 1976; Hasselström, 1981) et ULOAD (UMTA/FHWA, 1977). L'utilisation de ces logiciels est répandue chez les compagnies publiques de transport ainsi que les agences gouvernementales de planification.

Tous les algorithmes proposés jusqu'à date doivent être classifiés comme des approches heuristiques, car ils ne sont pas basés sur un principe de comportement du voyageur, donnant lieu à la formulation d'un modèle mathématique, qui est ensuite résolu par un algorithme. Il s'agit plutôt de variantes des techniques d'affectation qui proviennent du contexte du

transport routier (par exemple, chemin le plus court, affectation "multichemin"), adaptées pour tenir compte de quelques uns des phénomènes inhérents au réseau de transport en commun. Pour la plupart de ces méthodes, il n'est pas possible d'identifier le problème mathématique correspondant, et pour celles où c'est possible, on s'aperçoit qu'il s'agit d'une version restreinte du problème plus général qui sera élaboré ici, comprenant des contraintes additionnelles qui ne sont pas dues au comportement du voyageur mais uniquement à la méthode de calcul imposée par l'algorithme. Les solutions trouvées par ces méthodes ne sont donc que sous-optimales par rapport au problème général.

Les sections subséquentes de ce chapitre sont organisées de la façon suivante. Dans la section 1.2, nous établirons un modèle pour le comportement du voyageur de transport en commun qui nous amènera à la notion de la *stratégie optimale*. Ainsi pourrons-nous comparer entre elles les méthodes existantes et cerner les limites de chacun des algorithmes correspondants. Dans la section 1.3 on élaborera un modèle de programmation mathématique qui correspond au comportement postulé du voyageur qui suit la stratégie optimale. Nous démontrons que le problème peut s'exprimer en un programme linéaire dont la dimension croît exponentiellement avec la taille du réseau. Pour pouvoir rester dans une notation simple et générale, le problème est formulé sur un réseau généralisé, où chaque lien est soumis à un temps de déplacement et une attente. Le cas particulier où le temps d'attente ne dépend que de la fréquence combinée des liens considérés, sera examiné en détail dans la section 1.4. Nous verrons alors qu'on peut reformuler le problème comme un problème de programmation linéaire beaucoup plus simple, dont la dimension ne croît que linéairement avec la taille du réseau de transport en commun et qui ressemble fortement au problème des

chemins plus courts. Dans la section 1.5, nous présenterons un algorithme pour résoudre le problème ainsi redéfini, et nous verrons qu'il s'agit d'un algorithme à complexité polynomiale qui n'est que légèrement différent d'un algorithme des chemins les plus courts. Etant données ses exigences modestes en espace-mémoire et temps d'exécution requis, cet algorithme peut être appliqué même pour des réseaux de grande taille. Dans la section 1.6, le modèle est étendu au cas où les temps de déplacement sur les liens sont des fonctions croissantes du volume total du lien. Cela nous permettra de modéliser quelques aspects de la capacité finie d'un véhicule et de la congestion qui en résulte. On discutera les conditions d'équilibre qui en découlent et on développera un modèle de programmation convexe équivalent, ainsi qu'un algorithme de résolution efficace applicable à des problèmes de grande taille. La section 1.7 contient des résultats préliminaires sur la performance de l'algorithme proposé, obtenus à partir des réseaux de transport en commun de deux villes différentes, et ensuite on tirera des conclusions de cette étude et on indiquera les possibilités de recherches futures.

1.2 La Stratégie de choix de route optimale

Dans ce qui suit, nous considérons un réseau de transport en commun qui est constitué d'un ensemble de noeuds, d'un ensemble de lignes de transport en commun et d'un ensemble de liens piétonniers. L'itinéraire d'une ligne de transport en commun est défini comme une suite de noeuds qui correspond aux arrêts de la ligne, i.e., les endroits où les passagers peuvent monter et descendre d'un véhicule. Les temps de déplacement ou les coûts inhérents à chaque segment de ligne et à chaque lien piétonnier nous

sont connus; ils demeurent constants. Nous est aussi connue, pour chacun des noeuds desservis par une ligne, la distribution statistique des intervalles entre les arrivées de deux véhicules consécutifs de cette ligne. Par conséquent, si nous connaissons aussi le taux d'arrivée des passagers pour chaque noeud, nous pourrons établir la distribution du temps d'attente pour un vehicule d'une ligne donnée. De là, nous pourrons calculer l'espérance du temps d'attente combiné qui s'écoulera avant l'arrivée du premier véhicule, à l'intérieur de n'importe quel sous-ensemble de lignes se rencontrant en un noeud donné; nous pourrons aussi calculer la probabilité, pour chaque ligne parmi ce sous-ensemble, de voir un de ses véhicules y arriver le premier. On trouvera dans l'annexe A les détails de ces calculs.

Notons qu'on peut envisager un lien piétonnier comme une ligne de transport en commun qui ne relie que les deux noeuds adjacents au lien, qui a le même temps de déplacement que le lien piétonnier et qui ne comporte aucune attente (fréquence très élevée). Dans l'exposé qui suit, on ne parlera donc souvent que des lignes de transport en commun, sans tenir compte explicitement des liens piétoniers. Pour simplifier encore la formulation, nous supposons que le réseau de transport en commun est fortement connexe, i.e. que, de chaque noeud, on peut se rendre à tout autre noeud du réseau en utilisant un chemin formé de segments de lignes et de liens piétonniers. Quoique tous les résultats présentés plus bas restent valides même pour un reseau qui n'est pas fortement connexe, ou même non-connexe, on évitera de cette façon d'avoir à traiter des cas d'exception, qui, en soi, sont triviaux, mais nuisent à la clarté de l'exposé.

Observons maintenant notre voyageur. Son périple commence au noeud A

et a pour destination le noeud B. Nous prendrons pour acquis qu'il entend réduire au minimum l'espérance de la durée totale (ou du coût généralisé total) de son déplacement. Celle-ci est constituée de la somme, éventuellement la somme pondérée, du temps de marche, des périodes d'attente et des séjours à bord d'un véhicule. Cette hypothèse semble aller de soi et rejoint celle utilisée pour modéliser le choix de route des automobilistes. Dans le cas où les composantes du temps sont pondérées différemment, il est toujours possible de normaliser les facteurs de pondération des divers éléments du voyage à 1, par un simple changement d'échelle des temps correspondants. Il ne faut donc pas tenir compte explicitement de ces facteurs de pondération; on utilisera plutôt toujours le terme *temps*, même quand ces temps incluent en fait des facteurs de pondération et devraient, dans ce cas, être appelés des *coûts généralisés*.

A ce point, le lecteur a peut-être remarqué qu'on n'a pas considéré du tout les frais de déplacement, i.e. l'argent qu'on doit payer pour faire le voyage. Les frais ne sont pas considérés explicitement dans notre modèle, parce que dans la plupart des cas, ils ne dépendent pas du choix de route du passager. Ceci est vrai pour toutes les villes pour lesquelles le prix d'un voyage ne dépend que de l'origine et de la destination du voyage, mais non du chemin choisi entre les deux. Ceci est particulièrement vrai quand n'importe quel voyage coûte le même prix, qu'il s'agisse d'un voyage d'un coin de rue seulement, ou de traverser toute la ville en changeant plusieurs fois de ligne (en anglais "flat rate"), ou encore quand la plupart des passagers sont munis d'un abonnement mensuel ou hebdomadaire qui leur donne un accès non-restreint au réseau complet. Si par contre, le tarif d'un voyage dépend du chemin emprunté par le voyageur d'une façon additive (par exemple, quand le tarif est proportionnel à la distance

parcourue ou quand il dépend du nombre de fois qu'on a traversé une limite d'une zone de tarif), on peut ajouter ce tarif, avec une pondération appropriée, au temps de déplacement sur les segments. En général, le tarif est plutôt un élément qui influence le choix modal, mais qui n'a pas, ou que peu, d'influence sur le choix de route à l'intérieur du réseau de transport en commun.

On serait tenté de reformuler la question du choix de route du passager et de se demander: "Comment trouve-t-on le chemin du noeud A au noeud B qui réduise au minimum l'espérance du temps de voyage total?" Ainsi posé, le problème est déjà différent! On a l'habitude d'identifier l'ensemble des choix possibles (en anglais "choice set") auxquels se voit confronté le voyageur, désirant aller de A à B, avec l'ensemble des chemins distincts qui relient A et B. Tandis que cela tient pour l'automobiliste, ce n'est plus vrai dans le cas d'un passager du transport en commun. Pour diminuer le temps d'attente, celui-ci pourrait, par exemple, retenir un ensemble de chemins, puis attendre que l'arrivée du premier véhicule desservant un de ces chemins vienne décider lequel des chemins sera effectivement emprunté pour le voyage. Un tel choix ne correspond pas à choisir un chemin. L'ensemble des choix possibles est donc beaucoup plus grand que l'ensemble des chemins distincts: il inclut aussi des sous-ensembles de chemins. En limitant le problème à la recherche du chemin à temps minimum, on ne résout qu'une restriction du problème original, et, par conséquent, on n'arrive qu'à une solution sous-optimale.

Nous appellerons *stratégie* tout élément de l'ensemble de choix possibles qui s'offre au voyageur. Une stratégie, c'est encore un ensemble de règles dont l'application permet au voyageur d'atteindre sa destination.

Le nombre et la nature des diverses stratégies entre lesquelles le voyageur est appelé à faire son choix, dépendent de l'information qu'il peut acquérir pendant son voyage. Si, au cours de celui-ci, aucun renseignement supplémentaire ne lui parvient, "stratégie" devient simplement synonyme de "chemin". Pour le réseau qu'illustre la figure II-1-1, on aurait comme exemple d'une telle stratégie:

> "Prendre la ligne 2 jusqu'au noeud Y, correspondre avec la ligne 3, descendre au noeud B."

Mais si le voyageur, durant les périodes d'attente, apprend à quelle ligne appartiendra le prochain véhicule à passer, sa stratégie pourrait inclure des sous-ensembles de lignes et un exemple pourrait s'exprimer comme suit:

> "Prendre le prochain véhicule de la ligne 1 ou de la ligne 2; si je prends la ligne 1, descendre au noeud B; si je prends la ligne 2, correspondre au noeud Y avec un véhicule de la ligne 3 ou de la ligne 4 et descendre au noeud B."

Bien sûr, plus le voyageur est informé - sur le temps d'attente déjà écoulé, l'horaire d'arrivée des véhicules aperçus au cours de l'attente, l'identité d'autres véhicules qui sont visibles à travers les fenêtres du véhicule - et plus les stratégies peuvent devenir complexes. On aurait par exemple:

> "Attendre jusqu'à cinq minutes l'arrivée d'un véhicule de la ligne 1; à défaut, prendre la ligne 2; si, au noeud Y, on s'aperçoit de l'arrivée d'un véhicule de la ligne 3 (express), faire une correspondance à celui-ci; sinon continuer jusqu'au

noeud X pour y prendre, selon la première arrivée, la ligne 3 ou la ligne 4."

Dans tout le reste de ce chapitre, nous supposons que la seule information que le voyageur peut acquérir au cours de son périple, est qu'il connaît, pendant son attente, les lignes auxquelles appartiennent les véhicules qui s'approchent de l'arrêt. Pour chaque véhicule passant à l'arrêt, il peut alors décider, selon la ligne qui y correspond, s'il veut monter à bord de ce véhicule ou s'il veut attendre l'arrivée du prochain véhicule. Nous nous limitons donc aux stratégies qui correspondent aux deux premiers exemples cités plus haut. La définition d'une telle stratégie consiste alors à déterminer, pour chaque noeud du réseau sauf la destination, un ensemble non-vide de *lignes attrayantes*, et, pour chacune de celles-ci, le noeud où descend le voyageur. Les segments de ligne entre le noeud où le voyageur monte à bord du véhicule et le noeud où il descend seront appelés *segments actifs*. On remarquera que, sur un réseau comportant des liens piétonniers, une stratégie peut comporter un (seul) lien piétonnier sortant au lieu d'un ensemble de lignes attrayantes. Pour une stratégie donnée, le voyage se déroule alors selon un mécanisme qui peut s'exprimer comme ceci:

ETAPE 0: Soit NOEUD le noeud origine.

ETAPE 1: Monter à bord du premier arrivé de tous les véhicules qui relèvent des lignes attrayantes à NOEUD.

ETAPE 2: Descendre au noeud predéterminé.

ETAPE 3: Si on n'est pas encore rendu à la destination, NOEUD devient le noeud courant et on retourne à l'étape 1. Autrement, le voyage est terminé.

Une stratégie est *réalisable* si le graphe défini par les segments actifs des lignes attrayantes ne contient aucun cycle. Il faut se rappeler qu'une stratégie, selon cette définition, a un noeud destination bien déterminé et que, à l'opposé, le noeud d'origine n'entre pas dans la définition d'une stratégie. Une stratégie est donc un ensemble de règles qui permettent au voyageur d'atteindre sa destination, quelque soit son point de départ sur le réseau.

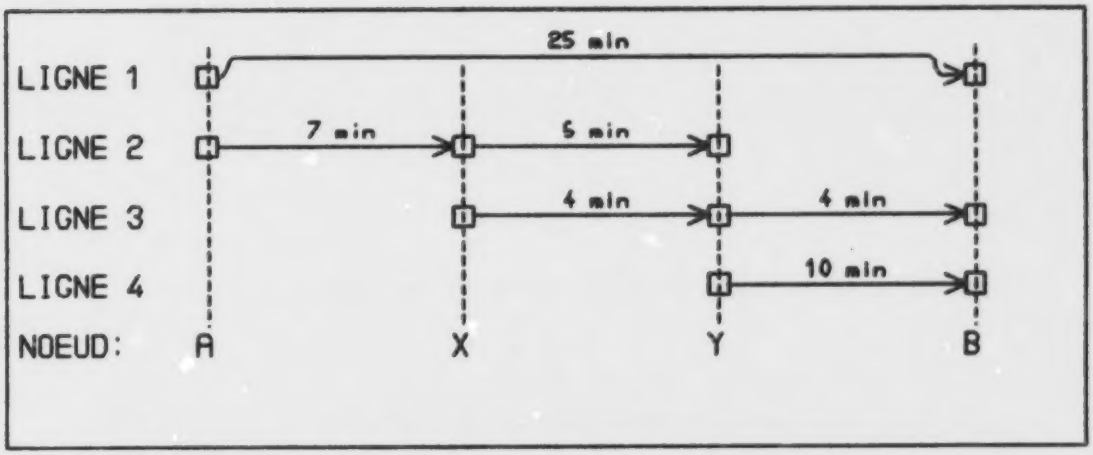

FIGURE II-1-1: Exemple d'un réseau de transport en commun.

La figure II-1-1, qui reprend les lignes de transport en commun précédemment utilisées pour illustrer la notion de stratégie, fait appel à un petit réseau, formé seulement de quatre noeuds et de quatre lignes. Le réseau ne contient aucun cycle; toutes les stratégies, au sens déjà défini, y seront réalisables. Il existe, en fait, 75 stratégies différentes pour atteindre le noeud B. On peut voir cela dans le tableau II-1-1 qui montre pour chaque noeud du réseau, à l'exception de la destination B, l'ensemble des sous-ensembles de lignes attrayantes possibles. Puisque chaque combinaison débouche sur une stratégie réalisable, il existe 5·5·3 = 75 de celles-ci pour atteindre B. Mais notons aussi qu'il n'y a que cinq chemins

differents entre A et B.

Noeud	Lignes attrayantes [Ligne -> Noeud]	Temps d'attente [Minutes]	Probabilités des lignes 1	2	3	4
A	1->B	6.0	1.00	-	-	-
A	2->X	6.0	-	1.00	-	-
A	2->Y	6.0	-	1.00	-	-
A	1->B. 2->X	3.0	0.50	0.50	-	-
A	1->B. 2->Y	3.0	0.50	0.50	-	-
X	2->Y	6.0	-	1.00	-	-
X	3->Y	15.0	-	-	1.00	-
X	3->B	15.0	-	-	1.00	-
X	2->Y. 3->Y	4.3	-	0.71	0.29	-
X	2->Y. 3->B	4.3	-	0.71	0.29	-
Y	3->B	15.0	-	-	1.00	-
Y	4->B	3.0	-	-	-	1.00
Y	3->B. 4->B	2.5	-	-	0.17	0.83

TABLEAU II-1-1: Tous les sous-ensembles de lignes attrayantes.

En prenant les temps de déplacement indiqués dans la figure II-1-1 et les espérances des temps d'attente, ainsi que les probabilités d'arriver premier de chaque ligne attrayante, telles que données dans le tableau II-1-1, il est possible de calculer l'espérance du temps de voyage total de A à B pour chaque stratégie. La figure II-1-2 montre la stratégie optimale, c'est-à-dire celle dont le temps de voyage est minimal. Notons que l'espérance du temps de voyage est de 27.75 minutes, et que cette stratégie ne correspond pas au choix d'un seul chemin. Les volumes v, indiqués sur les segments de ligne dans la figure II-1-2, correspondent à l'affectation d'un voyage de A à B selon la stratégie optimale.

La comparaison de cette stratégie avec celle qu'on obtient à l'aide des algorithmes basés sur la recherche du chemin le moins coûteux (Dial, 1967; Fearnside et Draper, 1971) - voir figure II-1-3 - montre la nette

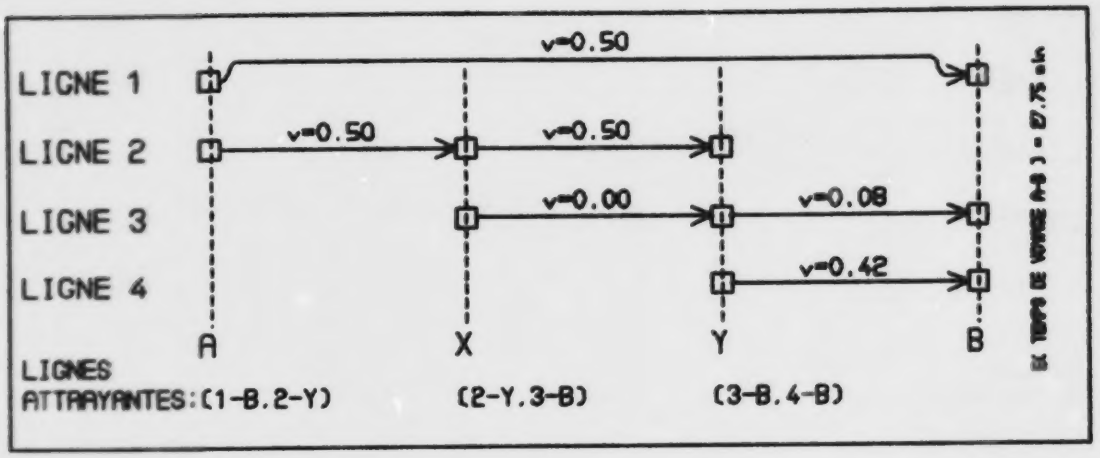

FIGURE II-1-2: La stratégie optimale.

infériorité de cette dernière. Rapp et al. (1976) décrivent aussi une méthode basée sur l'affectation des voyages aux chemins, mais le trafic est reparti sur tous les chemins "efficaces", selon la méthode d'affectation "multipath", qui a été proposée par Dial (1971) dans le contexte du transport privé. Puisque les temps d'attente considérés dans cette approche sont ceux des lignes individuelles (i.e., on ne considère pas de temps d'attente combiné), cette méthode donne lieu à une solution qui a des durées de voyages pires que celles qui correspondent aux chemins plus courts.

Il y a d'autres méthodes décrites dans la littérature (Le Clercq, 1972; Chriqui, 1974; Bolland, 1978), qui sont basées sur l'idée d'affecter les voyages sur les chemins plus courts; mais, au lieu de définir "chemin" comme une séquence de segments de lignes consécutifs, un chemin est défini comme une suite de noeuds de transfert. Entre deux noeuds de transfert, la

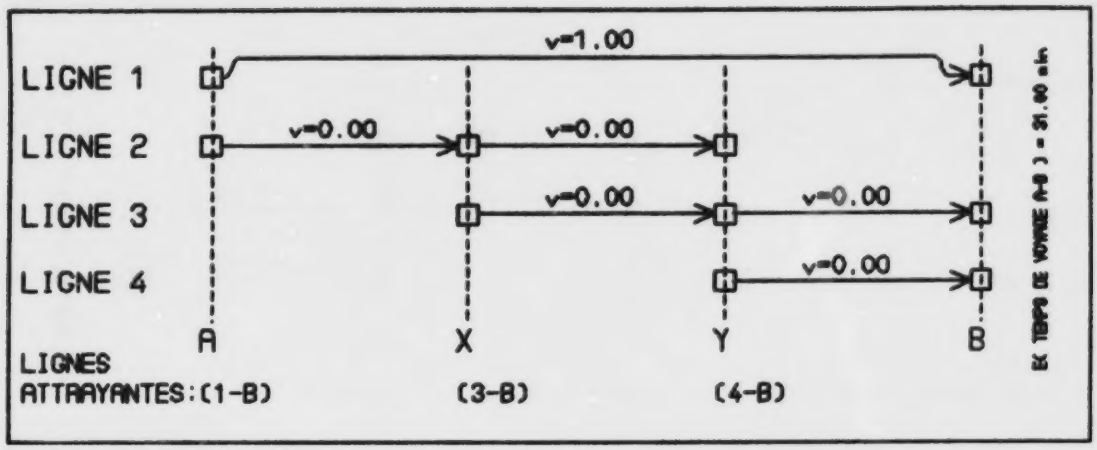

FIGURE II-1-3: Le chemin optimal.

stratégie du voyageur peut inclure plusieurs lignes différentes, pourvu qu'elles offrent un *service direct* entre les deux noeuds. Dans la méthode proposée par Le Clercq, toutes les lignes directes sont considérées attrayantes, tandis que Chriqui trouve la solution analytique pour le sous-ensemble optimal des lignes attrayantes à chaque noeud de transfert (Chriqui et Robillard, 1975). La solution qu'on trouve en appliquant l'algorithme de Chriqui - dans cet exemple, elle est identique à celle de l'algorithme de Le Clercq - est montrée dans la figure II-1-4. Bien que l'espérance du temps de voyage, 30.5 minutes, est meilleure que celle trouvée dans la figure II-1-3, elle est encore loin de la stratégie optimale. Bolland (1978) propose une méthode qui est aussi basée sur les chemins en terme de suite de noeuds de transfert; mais, au lieu d'affecter tous les voyages au chemin le plus court, plusieurs chemins sont considérés, selon des critères heuristiques.

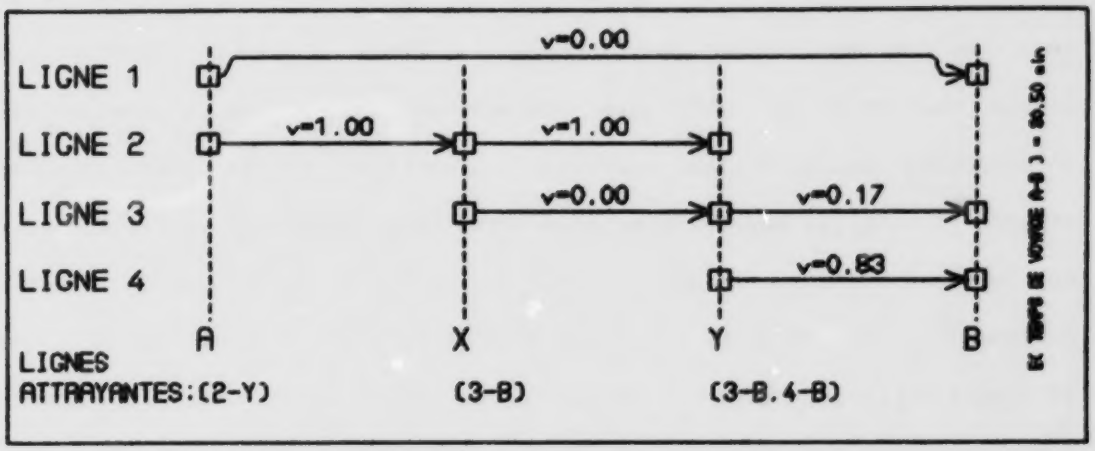

FIGURE II-1-4: La séquence optimale de noeuds de transfert.

Enfin, la méthode développée par Andreasson (1976) et ensuite raffinée par Hasselström (1981), est basée sur l'énumeration d'un certain nombre de chemins distincts reliant le noeud origine et le noeud destination. Parmi ceux-là, les chemins attrayants sont choisis à l'aide d'un ensemble complexe de règles heuristiques, qui, quoiqu'elles semblent donner des résultats satisfaisants en pratique, n'assurent pas l'optimalité de la solution en terme de temps de voyage. Contrairement aux autres méthodes qu'on a vues, qui fonctionnent toutes en affectant la demande d'une origine à toutes les destinations en un pas, cette méthode traite chaque paire origine-destination séparément; que cette méthode donne lieu à des programmes voraces en terme de temps de calcul ne nous étonne pas.

L'analyse des méthodes existantes, que nous venons de faire, était basée sur un petit exemple d'un réseau de transport en commun. On avait trouvé la stratégie optimale simplement en énumérant toutes les stratégies

possibles, ce qui n'est pas faisable dans le contexte d'un reseau de taille réaliste. De même, on supposait connus les temps d'attente combinés de chaque sous-ensemble de lignes attrayantes. Le but de cette étude est d'introduire un formalisme qui nous permet de traiter toutes les stratégies, telles que définies plus haut, et de caractériser la stratégie optimale en particulier,pour ensuite pouvoir formuler un modèle mathématique et développer un algorithme qui permette de trouver cette stratégie optimale. Comme on le verra, une fois qu'on pose les bonnes questions, le développement devient extrêmement simple, et dans les réponses qu'on trouve, on reconnaît des extensions naturelles de modèles bien connus.

1.3 Le modèle

Comme on l'a vu dans la section précédente, un voyage par transport en commun comporte plusieurs composantes; elles peuvent inclure entre autres:

- l'accès du noeud origine à un arrêt de ligne,
- l'attente d'un véhicule,
- la montée à bord d'un véhicule,
- le séjour dans un véhicule,
- la descente du véhicule,
- la marche d'un arrêt à un autre pour faire un transfert et
- l'accès de l'arrêt à la destination finale.

Nous utiliserons un temps (ou coût) non-négatif pour quantifier la durée de chacune de ces composantes. La seule exception est la composante d'attente, qui est quantifiée par la distribution statistique selon

laquelle sont distribués les temps d'attente pour l'arrivée du premier véhicule de la ligne au noeud donné.

Pour simplifier la présentation, nous proposons une formulation plus générale du problème; on ne traitera qu'un genre de composante qui inclut à la fois un coût et une attente. Ces composantes sont représentées par des liens $a \in A$ d'un réseau $G=(I,A)$ avec les noeuds $i \in I$. Soit A_i^+ (A_i^-) l'ensemble des liens sortant du (rentrant au) noeud $i \in I$. Chaque lien $a \in A$ est caractérisé par le tuple (c_a, G_a), où c_a dénote un coût non-négatif et G_a la fonction de distribution du temps d'attente, définie comme

$$G_a(x) = \text{Prob}\{\text{temps d'attente du lien } a \leq x\}.$$

La fonction $G_a(x)$ peut être calculée à partir de la distribution des intervalles de temps entre les arrivées de deux véhicules consécutifs et le taux d'arrivée des passagers à l'arrêt. Dénotons par $h_a(x)$ la densité de la distribution des intervalles de service sur le lien a, et avec H_a la distribution cumulative $H_a(x)=\int_0^x h_a(t)dt$. Si on suppose une distribution uniforme pour l'arrivée des passagers à l'arrêt, le temps d'attente pour service sur le lien a est distribué selon $g_a(x)=\dfrac{\int_0^x (1-H_a(t))dt}{\int_0^\infty (1-H_a(t))dt}$, avec distribution cumulative $G_a(x)=\int_0^x g_a(t)dt$.

Pour un lien a, qui correspond à une composante de voyage qui n'implique pas d'attente, $G_a(x)$ est définie comme

$$G_a(x) = \Delta(x) = \begin{cases} 0, & \text{si } x < 0, \\ 1, & \text{si } x \geq 0. \end{cases}$$

A l'aide de cette formulation généralisée, nous pouvons représenter tout réseau de transport en commun avec lignes à itinéraire fixe. Le mécanisme qui réalise cette transformation dépend fortement des particularités du réseau de transport en commun et du degré d'agrégation que l'on considère.

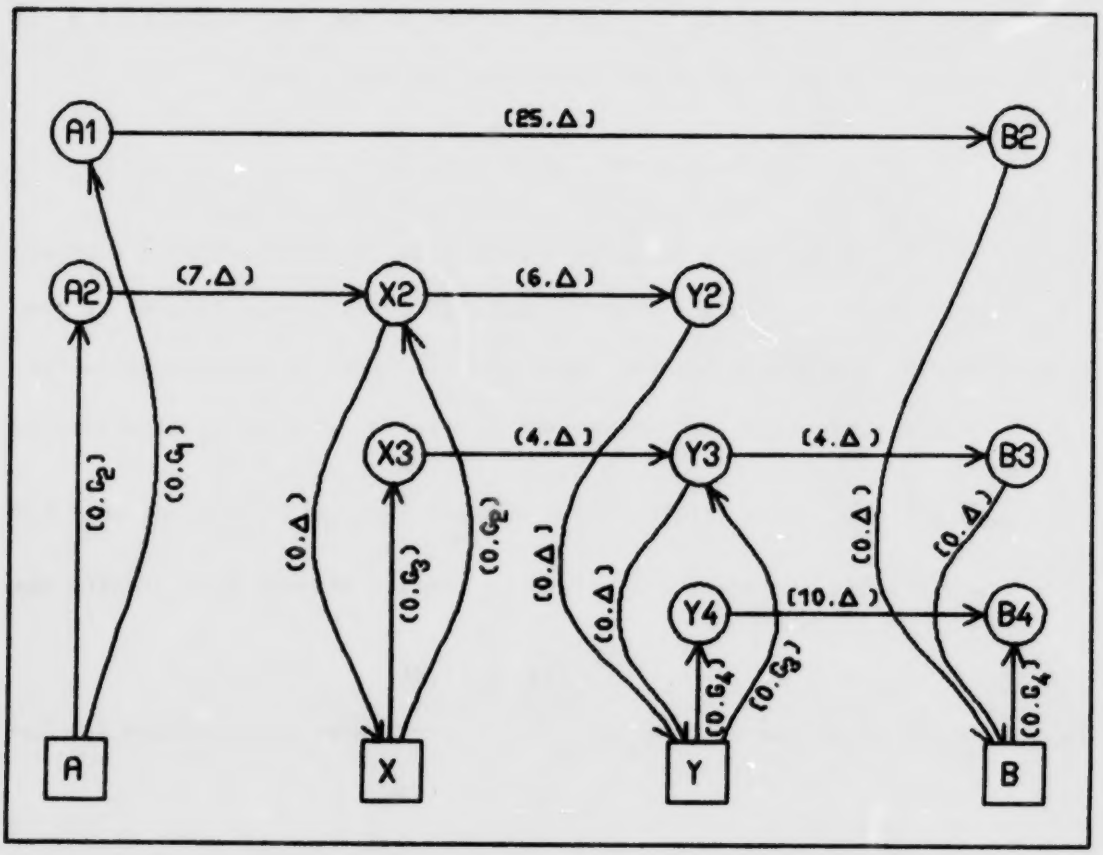

FIGURE II-1-5: Représentation "noeuds-liens" du réseau t.c.

La figure II-1-5 nous montre une telle représentation de réseau pour

l'exemple introduit dans la section 1.2. Dans ce cas-ci, nous n'avons que trois types de composantes du voyage: l'attente (pas de temps de déplacement), le séjour à bord du véhicule (pas d'attente) et la descente du véhicule (ni temps de déplacement ni d'attente). En notant qu'une suite de liens a_1 et a_2 avec (c_{a1}, G_{a1}) et $(c_{a2}, G_{a2} = \Delta)$ peut être remplacée par un seul lien avec les attributs $(c_a = c_{a1} + c_{a2}, G_a = G_{a1})$, le réseau de la figure II-1-5 peut être réduit à celui montré dans la figure II-1-6.

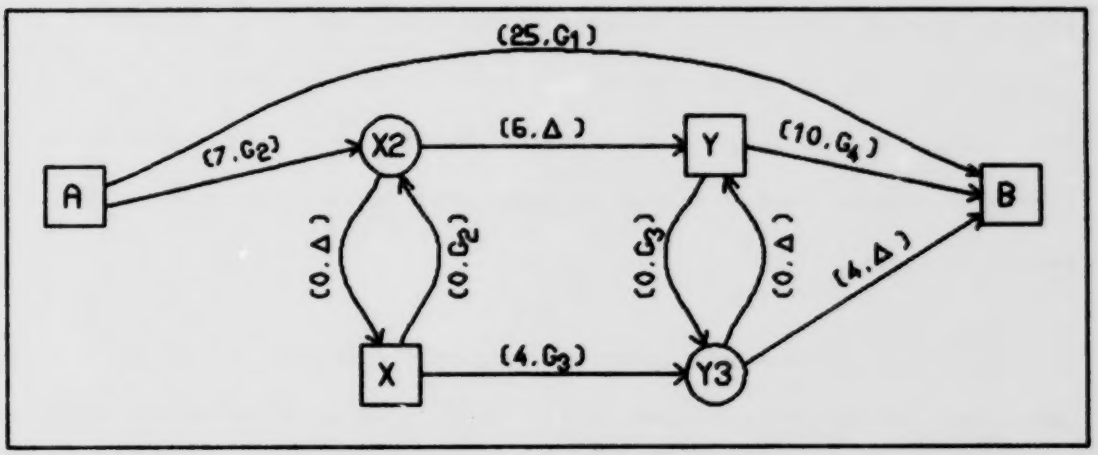

FIGURE II-1-6: Réseau simplifié.

Comme cette formulation généralisée ne dépend plus explicitement des lignes de transport en commun, - elle est définie seulement en termes de liens -, nous ne pourrons plus utiliser la terminologie associée avec les lignes, comme, par exemple, attendre l'arrivée d'un véhicule d'une ligne. On va donc adapter la notion qu'un lien est *desservi* par des véhicules, et que le passager, au lieu d'attendre l'arrivée d'un véhicule d'une certaine ligne, attend qu'un *lien* soit *desservi*.

Dans cette forme généralisée du problème de choix de route optimal en

transport en commun, une stratégie pour atteindre le noeud destination r est définie comme un réseau partiel $G_r=(I,\bar{A})$, qui ne contient que les liens qui correspondent aux segments actifs des lignes attrayantes. Nous pouvons alors dénoter une stratégie par le sous-ensemble des liens correspondants $\bar{A} \subseteq A$, ou encore, par la partition de ces liens par rapport à leur noeud initial ($\bar{A}_i^+ = A_i^+ \cap \bar{A}$, $i \in I$). Les liens qui ne font pas partie de la stratégie $\bar{A}$ ne sont jamais utilisés. Parmi les liens qui font partie de la stratégie $\bar{A}$ choisie par le voyageur, à chaque noeud $i \in I$, celui-ci monte à bord du premier véhicule qui dessert un des liens $a \in \bar{A}_i^+$. $\bar{A}_i^+$ correspond donc à l'ensemble des lignes attrayantes au noeud i, et bien sûr, on a que $\bar{A}_i^+ \neq \emptyset$ pour $i \neq r$. Une stratégie $\bar{A}$ est appelée *réalisable* s'il existe au moins un chemin de chaque noeud du réseau au noeud destination r et si $\bar{A}$ ne contient pas de circuits.

Soit $W(\bar{A}_i^+)$ l'espérance du temps d'attente combinée pour le premier véhicule qui dessert un des liens $a \in \bar{A}_i^+$. On appellera $W(\bar{A}_i^+)$ le temps d'attente combinée des liens $a \in \bar{A}_i^+$. De plus, soit $P_a(\bar{A}_i^+)$ la probabilité que le lien a soit desservi le premier parmi les liens $\bar{A}_i^+$. Il est commode de définir $P_a(\bar{A}_i^+)=0$ pour $a \notin \bar{A}_i^+$. Les deux quantités $W(\bar{A}_i^+)$ et $P_a(\bar{A}_i^+)$ peuvent être calculées à partir des distributions des temps d'attente $G_a(x)$:

$$W(\bar{A}_i^+) = \int_0^\infty \prod_{a \in \bar{A}_i^+} \{1-G_a(t)\}dt,$$

$$P_a(\bar{A}_i^+) = \int_0^\infty g_a(t) \prod_{a' \in \bar{A}_i^+ - (a)} \{1-G_{a'}(t)\}dt.$$

Le modèle qu'on présente ici est formulé pour affecter tous les voyages de tous les noeuds à un noeud spécifié r, appelé *destination*. Ceci

distingue ce modèle de la plupart des modèles d'affectation existants, soit dans le contexte routier ou en transport en commun, qui considèrent les voyages d'un noeud origine vers toutes les destinations à la fois. La demande g_i, $i \in I-\{r\}$, représente donc le nombre de voyages du noeud i à la destination r. Pour simplifier la notation, nous posons

$$g_r = - \sum_{i \neq r} g_i .$$

De plus, nous supposons $g_i > 0$, $i \in I-\{r\}$; ceci pour éviter certains phénomènes de dégénérescence de variables et donc permettre une analyse plus compacte du modèle. Nous soulignons par contre que tous les résultats présentés plus bas demeurent valides même dans le cas $g_i \geq 0$, $i \in I-\{r\}$.

Etant donnée une stratégie $\bar{A}$, la demande g_i, $i \in I$, est affectée au réseau pour obtenir des volumes de lien v_a, $a \in A$, et des volumes de noeud v_i, $i \in I$. Remarquons qu'il n'y a pas de confusion possible à utiliser le même nom de variable, v, pour désigner les volumes de lien et les volumes de noeud; l'appartenance de l'indice à l'ensemble A ou I indique d'une façon non-ambiguë s'il s'agit d'un volume de lien ou d'un volume de noeud. Le volume d'un noeud i, v_i, est la somme de tous les volumes des liens qui entrent au noeud plus la demande de ce noeud à la destination

$$v_i = \sum_{a \in A_i^-} v_a + g_i , \qquad i \in I. \tag{1}$$

Le volume de noeud v_i est redistribué sur les liens sortant de i qui appartiennent à la stratégie $\bar{A}$, selon leur probabilité d'être desservis le premier

$$v_a = P_a(\bar{A}_i^+)\, v_i , \qquad a \in A_i^+, \; i \in I. \tag{2}$$

Puisque $\sum_{a\varepsilon A_i^+} P_a(\bar{A}_i^+) = 1$, l'équation de conservation de flot

$$\sum_{a\varepsilon A_i^+} v_a - \sum_{a\varepsilon A_i^-} v_a = g_i \,, \qquad i\varepsilon I \tag{3}$$

est satisfaite à cause de (1) et (2).

La stratégie optimale $\bar{A}^*$ est celle qui minimise l'espérance du temps de voyage total, la somme des temps de déplacement et des temps d'attente. Elle est trouvée en résolvant le problème suivant pour v_i^*, $i\varepsilon I$, v_a^*, $a\varepsilon A$, et $\bar{A}^*$:

$$\text{Min} \quad \sum_{a\varepsilon A} c_a v_a + \sum_{i\varepsilon I} W(\bar{A}_i^+) v_i \tag{P1}$$

sujet à (1), (2) et

$$v_i \geq 0 \,, \qquad i\varepsilon I. \tag{4}$$

La non-négativité des volumes de lien v_a n'est pas énoncée explicitement; elle est une conséquence directe de (2) et (4).

Le problème P1 est formulé d'une façon inhabituelle: l'ensemble des variables de décision ne comprend pas seulement des variables numériques v_i et v_a, mais aussi le sous-ensemble inconnu $\bar{A}$. On peut reformuler le problème P1 en problème de programmation linéaire, qui ne dépend plus de l'ensemble $\bar{A}$. Pour faire cela, nous considérons les ensembles K_i, $i\varepsilon I$, soit les ensembles de tous les sous-ensembles possibles des liens sortant de i, $k=\bar{A}_i^+ \subseteq A_i^+$. Soit $w^k=W(k)$ et $p_a^k=P_a(k)$, $a\varepsilon A_i^+$, le temps d'attente combinée et les probabilités des liens correspondant au sous-ensemble $k\varepsilon K_i$. Les volumes de noeud v_i sont partitionnés en composantes v^k, $k\varepsilon K_i$, une composante pour chaque sous-ensemble de liens sortant d'un noeud. Cela nous

permet de formuler le problème de programmation linéaire

$$\text{Min} \quad \sum_{a \in A} c_a v_a + \sum_{i \in I} \sum_{k \in K_i} \omega^k v^k \qquad \text{(P2)}$$

sujet à

$$\sum_{k \in K_i} v^k = \sum_{a \in A_i^-} v_a + g_i , \qquad i \in I, \qquad (5)$$

$$v_a = \sum_{k \in K_i} p_a^k v^k , \qquad a \in A_i^+, i \in I, \qquad (6)$$

$$v^k \geq 0 , \qquad k \in K_i, i \in I. \qquad (7)$$

Proposition 1: Les problèmes P1 et P2 sont équivalents.

Preuve: En premier lieu, nous notons qu'un point extrémal de P2 est une solution réalisable qui comporte au plus $|I|-1$ de k avec $v^k>0$. Chaque solution réalisable $(v,\bar{A})$ du problème P1 correspond donc à un point extrême du problème P2, de la forme

$$v_a' = v_a , \qquad a \in A,$$

$$v^k = \begin{cases} v_i & \text{si } k=\bar{A}_i^+ \\ 0 & \text{sinon} \end{cases} , \qquad k \in K_i, i \in I.$$

et donne lieu à la même valeur de la fonction objective. Dans le problème P2, (6), combiné avec la positivité des demandes g_i, implique qu'il existe au moins un $k \in K_i$ avec $v^k>0$ pour chaque noeud $i \in I-(r)$. Chaque point extrémal correspond donc à une solution réalisable de P1. La domaine

réalisable de P2 est donc l'enveloppe convexe des points réalisables de P1. Puisqu'un programme linéaire est toujours équivalent à sa restriction aux points extrémaux, P2 est équivalent à P1.■

Après avoir transformé le problème original en problème de programmation linéaire, il est naturel d'examiner sa formulation duale

$$\text{Max} \quad \sum_{i \in I} g_i u_i \tag{P3}$$

sujet à

$$u_j + c_a + \mu_a = u_i \ , \qquad a=(i,j) \in A, \tag{8}$$

$$\sum_{a \in A_i^+} p_a^k \mu_a \leq \omega^k \ , \qquad k \in K_i, i \in I. \tag{9}$$

Les variables duales u_i correspondent à l'espérance du temps de voyage total pour atteindre la destination r à partir de l'origine i. Cette interprétation devient évidente en regardant la fonction objective duale, qui, à l'optimum, a la même valeur que la fonction objective du problème primal P2. Puisque P3 est invariant sous des translations de type $u_i' = u_i + c$, $i \in I$, nous pouvons toujours supposer $u_r = 0$ sans perte de généralité.

Proposition 2: Toute solution optimale (u^*, μ^*) du problème P3 correspond à une stratégie $\bar{A}^*$ qui satisfait

$$\sum_{a \in A_i^+} P_a(\bar{A}_i^{+*}) \mu_a^* = W(\bar{A}_i^{+*}) \ , \qquad i \in I. \tag{10}$$

Preuve: Evidemment, l'équation (10) est satisfaite pour toute solution optimale de P1, comme on peut voir facilement en appliquant le théorème des écarts complémentaires de la programmation linéaire aux problèmes P2 et P3,

qui implique

$$\sum_{a \in \bar{A}_i^+} p_a^k \mu_a^* = w^k \quad \text{si } v^{k^*} > 0, \qquad i \in I.$$

En appliquant la proposition 1, on est assuré de l'existence de $\bar{A}^*$. ■

Les variables duales μ_a n'ont pas d'interprétation immédiate. Malgré tout, l'équation (10) nous indique que les variables μ_a constituent des composantes du temps d'attente, distribué parmi les liens $a \in \bar{A}_i^+$; leur somme, pondérée par p_a^k, est égale au temps d'attente combinée $W(\bar{A}_i^+)$. Une borne supérieure pour les variables μ_a suit directement de (9)

$$\mu_a \leq W(a) , \qquad a \in A, \qquad (11)$$

car chaque lien constitue lui-même un sous-ensemble de $\bar{A}_i^+$ et correspond donc à un $k \in K_i$.

Nous soulignons que les variables μ_a, $a \in A$ ne correspondent pas à l'espérance du temps d'attente, conditionnelle au fait que le lien a a été desservi en premier parmi les liens $\bar{A}_i^+$.

1.4 Le cas spécial des fréquences combinées

Dans la section suivante, nous étudierons en détail le cas spécial, pour lequel la distribution du temps d'attente pour chaque lien (ou ligne de transport en commun, dans le problème original) est quantifiée à l'aide d'un seul paramètre f_a, qu'on appellera la *fréquence* du lien a. Le temps d'attente combinée et les probabilités des liens sont obtenus à partir des fréquences selon les formules suivantes:

$$W(\bar{A}_i^+) = \frac{\alpha}{\sum_{a \in \bar{A}_i^+} f_a}, \qquad \alpha > 0, \tag{12}$$

$$P_a(\bar{A}_i^+) = \frac{f_a}{\sum_{a' \in \bar{A}_i^+} f_{a'}}, \qquad a \in \bar{A}_i^+. \tag{13}$$

Il est facile de vérifier que, si on met $\alpha=1$, on arrive au cas où les intervalles entre deux véhicules consécutifs sont distribués selon une loi exponentielle avec moyenne $1/f_a$.

Le cas $\alpha=1/2$ est une approximation d'un horaire à intervalles fixes de $1/f_a$ pour chaque lien a. Cette mesure du temps d'attente est de loin celle qui est utilisée le plus souvent en pratique (Chapleau, 1974; Dial, 1967; Le Clercq, 1972; Rapp et al., 1976; UMTA/FHWA, 1977; Bolland J.D., 1978), malgré le fait qu'elle ne constitue qu'une approximation grossière.

De plus, le facteur α peut inclure l'effet de la perception différente du temps passé à attendre ou à bord d'un véhicule. Puisque les probabilités des liens, telles que définies dans (13), ne dépendent pas de l'unité choisie pour les fréquences f_a, et le temps d'attente combinée (12) en dépend d'une façon linéaire, il est toujours possible de changer l'échelle des fréquences par un facteur de $1/\alpha$. Sans perte de généralité, nous pouvons donc supposer que $\alpha=1$ pour le développement suivant.

Pour pouvoir exploiter la structure particulière du cas spécial que l'on considère, (12) et (13), il nous faut exprimer une stratégie $\bar{A}$ en termes des variables 0-1 x_a

$$x_a = \begin{cases} 0 & \text{,si } a \in \bar{A} \\ & \\ 1 & \text{,si } a \in \bar{\bar{A}} \end{cases} \quad , \qquad a \in A. \tag{14}$$

Cela nous permet de reécrire Pi pour le cas où les temps d'attente et les probabilités des liens satisfont (12) et (13).

$$\text{Min} \quad \sum_{a \in A} c_a v_a + \sum_{i \in I} \frac{v_i}{\sum_{a \in A_i^+} f_a x_a} \tag{P4}$$

sujet à

$$v_a = \frac{x_a f_a}{\sum_{a' \in A_i^+} f_{a'} x_{a'}} v_i \quad , \qquad a \in A_i^+, i \in I, \tag{15}$$

$$v_i = \sum_{a \in A_i^-} v_a + g_i \quad , \qquad i \in I, \tag{16}$$

$$v_i \geq 0 \quad , \qquad i \in I, \tag{17}$$

$$x_a = 0 \text{ ou } 1 \quad , \qquad a \in A. \tag{18}$$

En comparant le problème P4 avec P2 et P3, nous constatons que nous avons déjà réduit de beaucoup la dimension du problème à résoudre. Le nombre de variables et le nombre de contraintes sont maintenant proportionnels au nombre de noeuds et de liens dans le réseau. L'énumération explicite de tous les sous-ensembles de liens attrayants n'est plus requise. Par contre, le problème en soi semble être devenu beaucoup plus difficile à

résoudre. A première vue, le problème P4 consiste en une fonction objective non-linéaire et même non-convexe, sujette à des contraintes non-convexes et contenant aussi des variables binaires. Comme on démontrera à l'aide du développement qui suit, il est possible de réduire ce problème à un simple problème de programmation linéaire, sans augmenter sa taille.

D'abord, nous notons que les contraintes de non-négativité sur les volumes de noeuds (17) peuvent être remplacées par des contraintes de non-négativité sur les volumes de liens

$$v_a \geq 0 , \qquad a \in A; \qquad (19)$$

L'équation (16) et la positivité de la demande montrent que (19) implique (17); d'autre part, l'équation (15) nous assure que (17) implique (19).

Ensuite, en prenant la somme des contraintes (15), nous obtiendrons

$$\sum_{a \in A_i^+} v_a = v_i , \qquad i \in I, \qquad (20)$$

ce qui nous amène, en substituant dans (16), aux contraintes de conservation de flôt (3).

Nous pouvons maintenant introduire des nouvelles variables w_i, qui dénoteront le temps d'attente total de tous les voyages qui utilisent le noeud i.

$$w_i = \frac{v_i}{\sum_{a \in A_i^+} f_a x_a} , \qquad i \in I, \qquad (21)$$

Avec (21) on peut substituer les variables v_i dans la fonction objective et dans (15). On a donc sustitué les trois occurrences de la variable v_i dans le problème P4. Cela démontre donc la proposition suivante:

Proposition 3: Le problème P4 est équivalent à

$$\text{Min} \quad \sum_{a \in A} c_a v_a + \sum_{i \in I} w_i \tag{P5}$$

sujet à (3), (18), (19) et

$$v_a = x_a f_a w_i \,, \qquad a \in A_i^+, i \in I. \tag{22}$$

Comme on peut le voir, la fonction objective est devenue linéaire et les volumes des noeuds, v_i, $i \in I$, n'apparaissent plus dans la formulation. Les variables 0-1, x_a, ne sont utilisées que dans les contraintes (22), qui, en même temps, constituent les seules contraintes non-linéaires du problème P5. En appliquant une relaxation aux contraintes (22), soit

$$v_a \leq f_a w_i \,, \qquad a \in A_i^+, i \in I, \tag{23}$$

on obtient un problème de programmation linéaire

$$\text{Min} \quad \sum_{a \in A} c_a v_a + \sum_{i \in I} w_i \tag{P6}$$

sujet à (3), (19) and (23), comme suit

$$v_a \leq f_a w_i \,, \qquad a \in A_i^+, i \in I, \tag{23}$$

$$\sum_{a \in A_i^+} v_a - \sum_{a \in A_i^-} v_a = q_i \,, \qquad i \in I, \tag{3}$$

$$v_a \geq 0 \,, \qquad a \in A; \tag{19}$$

Puisqu'on a obtenu P6 comme une relaxation de P5, il faut prouver la proposition suivante:

Proposition 4: Le problème P6 est équivalent au problème P5.

Preuve: P6 comporte $2|A|+|I|-1$ contraintes indépendantes et $|A|+|I|-1$ variables (w_r est toujours zéro). Pour chaque point extrémal de la région réalisable du problème P6, il y a donc *au plus* $|A|$ contraintes d'inégalité qui sont satisfaites d'une façon stricte. Puisqu'il y a deux contraintes d'inégalité associées avec chaque lien, (19) et (23), qui, on le remarque, ne peuvent pas être satisfaites en même temps (mutuellement exclusives), il s'en suit qu'il y a *au moins* une inégalité stricte associée avec chaque lien. On peut donc conclure que, pour chaque lien $a\varepsilon A$, exactement une des deux contraintes, (19) ou (23), doit être satisfaite avec égalité; ce qui veut dire que (23) est satisfait. Cela démontre que chaque point extrême de P6 est un point réalisable de P5. Puisque P6 est une relaxation de P5, cela démontre que les deux problèmes sont équivalents. ■

Une fois qu'on obtient le modèle de programmation linéaire P6, il est possible d'étudier sa formulation duale, soit

$$\text{Max} \quad \sum_{i\varepsilon I} g_i u_i \tag{P7}$$

sujet à

$$u_j + c_a + \mu_a \geq u_i \,, \qquad a=(i,j)\varepsilon A, \tag{24}$$

$$\sum_{a\varepsilon A_i^+} f_a \mu_a = 1 \,, \qquad i\varepsilon I, \tag{25}$$

$$\mu_a \geq 0 \,, \qquad a\varepsilon A. \tag{26}$$

Soient (v^*,w^*) et (u^*,μ^*) des solutions optimales quelconques des problèmes P6 et P7. Le théorème des écarts complémentaires en programmation linéaire nous assure que les conditions

$$(v_a^* - f_a w_i^*)\,\mu_a^* = 0\,, \qquad a\in A_i^+, i\in I, \tag{28}$$

et

$$(u_j^* + c_a + \mu_a^* - u_i^*)\,v_a^* = 0\,, \qquad a=(i,j)\in A. \tag{29}$$

sont toujours satisfaites.

Jusqu'ici, nous n'avons considéré que le cas où chaque lien implique un temps d'attente non-nul, i.e. une fréquence finie y est associée. Si, pour un sous-ensemble de liens $\underline{A} \subseteq A$, aucune attente n'est impliquée, le développement ci-haut n'est pas applicable tel quel. Il y a deux façons de traiter ce cas. La première façon consiste simplement à mettre $f_a=M$, où M dénote un nombre suffisamment grand pour que le temps d'attente 1/M soit considéré comme négligeable. Une autre façon de traiter les liens sans attente est de modifier légèrement les problèmes P4, P5 et P6. Pour les problèmes P4 et P5, cela résulte dans l'élimination des contraintes (22) et (23) pour $a\in\underline{A}$ et, en même temps, dans l'ajout des contraintes de non-négativité pour w_i, $i\in I$. Dans le problème dual, P6, la contrainte (24) ne s'applique qu'aux liens $a\notin\underline{A}$ et elle est remplacée par $u_j+c_a \geq u_i$ pour $a\in\underline{A}$. La contrainte (25) devient

$$\sum_{a\in\underline{A}_i^+} f_a\mu_a \leq 1\,, \qquad i\in I.$$

Les conditions d'écarts complémentaires (28) et (29) sont réduites à $(u_j^*+c_a-u_i^*)v_a^*=0$ pour $a\in\underline{A}$.

Notons enfin que le modèle qu'on vient de développer, dans la formulation primale comme dans la formulation duale, ressemble fortement au problème d'affectation "tout-ou-rien" sur les chemins les plus courts. Plus précisément, ce dernier correspond au cas particulier du problème d'affectation pour le transport en commun, où aucun des liens n'est sujet à l'attente, soit où $\underline{\theta} = A$. On ne sera donc pas surpris d'apprendre que, comme on le verra dans la section suivante, cette ressemblance s'étend même aux algorithmes de résolution pour les deux problèmes.

1.5 Un algorithme de résolution

Dans cette section, nous présenterons un algorithme de résolution pour le problème d'affectation pour le transport en commun P4. L'algorithme est composé de deux parties. Dans une première passe à travers le réseau on calcule la stratégie optimale $\bar{A}^*$ pour se rendre à la destination à partir de n'importe quel noeud origine, ainsi que les espérances des temps de voyage u_i^*, $i \in I$. Dans la deuxième partie, les voyages de tous les noeuds origines à la destination sont chargés sur le réseau selon la stratégie optimale.

Voici l'énoncé de l'algorithme:

Partie 1: Trouver la stratégie optimale

1.1 (Initialisation) $u_i := \infty$, $i \in I-(r)$; $u_r := 0$;

$f_i := 0$, $i \in I$;

$S := A$; $\bar{A} := \emptyset$.

1.2 (Recherche du prochain lien) Si $S = \emptyset$ alors STOP,

sinon trouver $a=(i,j)\in S$ qui satisfait

$$u_j + c_a \leq u_{j'} + c_{a'}, \quad a'=(i',j')\in S;$$

$S := S - (a)$.

1.3 (Mise à jour des étiquettes) Si $u_i \geq u_j + c_a$ alors

$$u_i := \frac{f_i u_i + f_a (u_i + c_a)}{f_i + f_a},$$

$f_i := f_i + f_a$, $\bar{A} := \bar{A} + (a)$;

aller à l'étape 1.2.

Partie 2: Affecter la demande selon la stratégie optimale

2.1 (Initialisation) $v_i := g_i$, $i\in I$;

2.2 (Chargement) Pour chaque lien $a\in A$, en ordre décroissant de $(u_j + c_a)$ faire:

si $a\in\bar{A}$ alors $v_a := \frac{f_a}{f_i} v_i$,

$v_j := v_j + v_a$,

sinon $v_a := 0$.

L'étape 1.1 initialise les étiquettes u_i, i.e. les espérances du temps de voyage de i à la destination r. Les variables auxiliaires f_i, $i\in I$, initialement mises à zéro, contiendront les fréquences combinées des liens choisis dans la stratégie optimale au noeud i. L'ensemble S identifie les liens qui n'ont pas encore été examinés. L'ensemble $\bar{A}$ contiendra la stratégie optimale, i.e. le sous-ensemble des liens attrayants.

L'étape 1.2 choisit le lien a=(i,j) le plus proche de la destination, qui n'a pas encore été examiné. La mesure de proximité considérée est u_j+c_a, c'est-à-dire l'espérance du temps de voyage du début du lien à la destination, sans compter le temps d'attente au noeud i. Si ce temps est plus petit que l'étiquette courante de i, u_i, le lien a est inclus dans la stratégie optimale, et l'étiquette u_i et la fréquence combinée f_i sont mises à jour selon les formules données dans l'étape 1.3. Il faut noter ici, que, lorsque l'étiquette d'un noeud est modifiée pour la première fois, on a $f_i=0$ et $u_i=\infty$. Pour garder l'énoncé de l'algorithme le plus compact possible, nous adapterons la convention $0\cdot\infty = 1$. (Pour $\alpha\neq 1$ dans (12), cette convention devient $0\cdot\infty = \alpha$). La première partie de l'algorithme se termine quand tous les liens ont été examinés.

La deuxième partie de l'algorithme affecte la demande g_i, $i\in I-\{r\}$, sur le réseau, selon la stratégie $\bar{A}$ trouvée dans la première partie. Cela est fait en chargeant chaque lien $a\in\bar{A}$ avec la proportion du volume du noeud i, v_i, qui correspond à sa fréquence. Puisque les liens sont traités en ordre topologique inverse (u_j+c_a décroissant), les volumes de noeud peuvent être mis à jour en parallèle; la procédure n'examine donc chaque lien qu'une seule fois.

Après avoir donné un aperçu général de l'algorithme proposé, nous procèderons maintenant à la preuve d'optimalité, i.e., on démontrera qu'en effet, cet algorithme trouve la solution optimale du problème P4. Puisque l'algorithme calcule à la fois les variables duales u_i, $i\in I$, et les variables primales v_a, $a\in A$, on n'a qu'à vérifier la réalisabilité des solutions primales et duales respectives, ainsi que les conditions des écarts complémentaires. Pour faire cela, nous dénoterons $(\hat{u},\hat{v})$ la solution

trouvée par l'algorithme.

Proposition 5: $\hat{u}$ correspond à une solution réalisable du problème dual P7.

Preuve: Soient u_i^a et u_j^a les valeurs des variables u_i et u_j, immédiatement après que l'étape 1.3 ait été appliquée au lien a=(i,j). Bien sûr, u_i^a et u_j^a satisfont

$$\begin{aligned} u_i^a &> u_j^a + c_a \quad \text{,si } a\in\bar{A}, \\ u_i^a &\leq u_j^a + c_a \quad \text{,sinon.} \end{aligned} \tag{30}$$

Il n'est pas possible que l'étiquette u_j soit encore modifiée après que l'étape 1.3 ait été appliquée au lien (i,j), car les liens sont examinés en ordre croissant de u_j+c_a et on exige $c_a\geq 0$. On a donc $u_j^a=\hat{u}_j$, $a=(i,j)\in A$. Le même raisonnement ne tient pas pour u_i, qui peut encore être modifié après que le lien (i,j) ait été traité. Par contre, on sait que les étiquettes ne peuvent que décroître, parce que que la nouvelle valeur de u_i est une combinaison convexe de l'ancienne valeur de u_i et d'une valeur plus petite, soit u_j+c_a. On a donc $\hat{u}_i \leq u_i^a$, $a=(i,j)\in A$. Toutefois, il existe un lien $a'\in A$ qui satisfait $\hat{u}_i=u_i^{a'}$; c'est le lien qui a été inclu comme dernier dans $\bar{A}_i^+$. En utilisant (30), cela nous amène à

$$\hat{u}_i > u_j^{a'} + c_{a'} \geq u_j^a + c_a , \qquad a\in\bar{A}_i^+, \tag{31}$$

ce qui prouve que $\hat{u}$ satisfait

$$\begin{aligned} \hat{u}_i &> \hat{u}_j + c_a \quad \text{,si } a\in\bar{A}, \\ \hat{u}_i &\leq \hat{u}_j + c_a \quad \text{,sinon.} \end{aligned} \tag{32}$$

Comme u_j ne changera plus sa valeur après qu'on l'ait utilisé dans l'étape 1.3, on peut exprimer $\hat{u}_i$ en termes des $\hat{u}_j$ de la façon suivante:

$$\hat{u}_i = \frac{\sum_{a\in\bar{A}_i^+} f_a(\hat{u}_j+c_a) + 1}{\sum_{a\in\bar{A}_i^+} f_a}, \qquad i\in I. \tag{33}$$

Dans (33), on reconnait le premier terme comme étant la moyenne pondérée des temps de voyages de chaque lien à la destination et le deuxième terme est l'espérance du temps d'attente au noeud i. (Remarquons ici que le "1" provient de notre convention de mettre $0\cdot\infty=1$, lors de la première modification de u_i.) Nous construisons les variables duales $\hat{\mu}$ comme

$$\hat{\mu}_a = \begin{cases} \hat{u}_i-\hat{u}_j-c_a & ,\text{si } a\in\bar{A}, \\ 0 & ,\text{sinon.} \end{cases} \tag{34}$$

En réarrangeant (33), on obtient

$$\sum_{a\in\bar{A}_i^+} f_a(\hat{u}_i-\hat{u}_j-c_a) = 1, \qquad i\in I, \tag{35}$$

et en insérant (34) dans (35)

$$\sum_{a\in A_i^+} f_a\hat{\mu}_a = 1, \qquad i\in I. \tag{36}$$

Cela démontre que $(\hat{u},\hat{\mu})$ satisfait (24) et (26), il s'agit donc d'une solution réalisable du problème dual P7. ■

Proposition 6: $\hat{v}$ correspond à une solution réalisable du problème primal P6.

Preuve: Soit $\hat{f}_i$, $i\in I$, les fréquences combinées qui résultent de la partie 1 de l'algorithme. Selon l'étape 1.3, elles satisfont

$$\hat{f}_i = \sum_{a\in\bar{A}_i^+} f_a, \qquad i\in I. \tag{37}$$

Nous pouvons construire les variables $\hat{w}_i$, $i \in I$, comme

$$\hat{w}_i = \frac{\hat{v}_i}{\hat{f}_i}, \qquad i \in I. \tag{38}$$

Les variables $\hat{v}_a$ peuvent être exprimées en termes de $\hat{w}_i$ selon les formules de l'étape 2.2 de l'algorithme

$$\hat{v}_a = \begin{cases} f_a \hat{w}_i & \text{,si } a \in \bar{A}, \\ 0 & \text{,sinon.} \end{cases} \tag{39}$$

Puisque, par construction, les volumes satisfont les contraintes de conservation de flôt, on peut conclure que $(\hat{v},\hat{w})$ est une solution réalisable du problème primal P6. En plus, la solution correspond à une stratégie $\bar{A}$; elle constitue donc un point extrémal de la région réalisable de P6. ■

Proposition 7: $(\hat{u},\hat{v})$ sont optimaux.

Preuve: Les liens qui appartiennent à la stratégie $\bar{A}$ satisfont, selon (39),

$$\hat{v}_a - f_a \hat{w}_i = 0, \qquad a=(i,j) \in \bar{A}, \tag{40}$$

et selon (34)

$$\hat{u}_j + c_a + \hat{\mu}_a - \hat{u}_i = 0, \qquad a=(i,j) \in \bar{A}. \tag{41}$$

Par contre les liens qui ne font pas partie de la stratégie $\bar{A}$ satisfont

$$\hat{v}_a = 0, \quad \hat{\mu}_a = 0, \qquad a \notin \bar{A}. \tag{42}$$

Cela démontre que $(\hat{v},\hat{w})$ et $(\hat{u},\hat{\mu})$ satisfont les conditions des écarts complémentaires (28) et (29); ils constituent donc des solutions optimales des problèmes P6 et P7. Puisqu'en plus, $(\hat{v},\hat{w})$ est une solution de base (point extrémal) de P6, les propositions 3 et 4 nous indiquent que $(\hat{v},\bar{A})$

est une solution optimale du problème P4. ■

Nous revenons ici brièvement au cas où un sous-ensemble de liens $\underline{A} \subseteq A$ ne comporte pas de temps d'attente. On peut encore traiter ce cas simplement en mettant les fréquences de ces liens à un grand nombre, ou, comme alternative, adapter l'algorithme à ce cas particulier. Cette dernière approche nous amène à deux petites modifications dans les étapes 1.3 et 2.2 de l'algorithme. Pour $a \in \underline{A}$, la mise à jour des étiquettes dans l'étape 1.3 devient

$$u_i := u_j + c_a ,$$
$$f_i := \infty , \quad \bar{A}_i^+ := \{a\}.$$

Dans l'étape 2.2, le volume d'un lien sans attente $a \in \underline{A}$ qui fait partie de la stratégie optimale, est simplement $v_a := v_i$. On ne donnera pas de preuve d'optimalité pour cette version modifiée de l'algorithme, parce qu'elle suit les mêmes lignes que les preuves des propositions 5, 6 et 7.

Pour illustrer le fonctionnement de l'algorithme, nous l'appliquerons au même petit réseau de transport en commun qu'on a déjà utilisé dans les sections 1.2 et 1.3 et qui est montré dans la figure II-1-1. Comme on le voit maintenant, les temps et les probabilités des liens qu'on s'est donné dans le tableau II-1-1, correspondent effectivement au cas spécial (12) et (13). Il est possible de spécifier des fréquences sur les liens telles qu'elles reproduisent les temps d'attente et les probabilités des liens du tableau II-1-1. La figure II-1-7 montre le réseau simplifié en indiquant sur chaque lien son temps de déplacement et sa fréquence dans la forme (c_a, f_a).

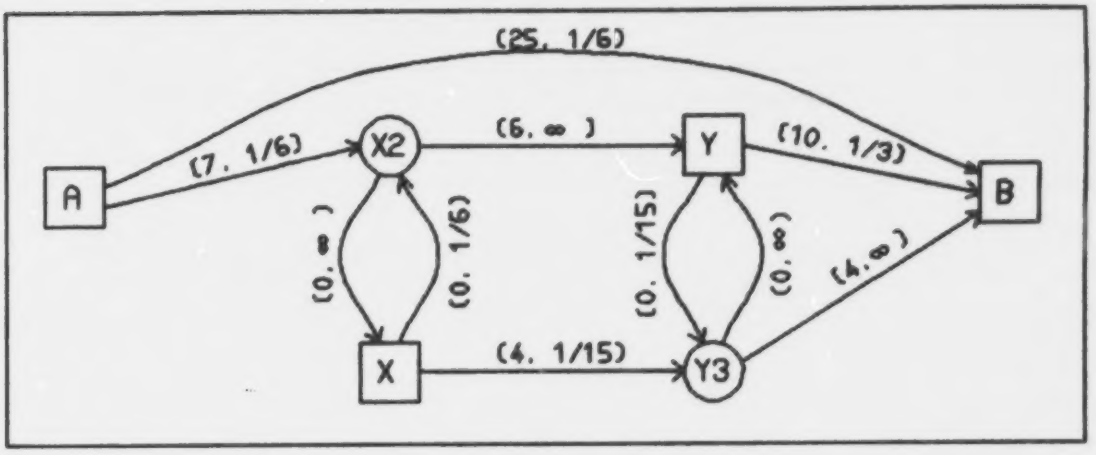

FIGURE II-1-7: Réseau avec temps et fréquences sur les liens.

Puisque cet exemple contient des liens sans attente ($f_a = \infty$), la version modifiée de l'algorithme est utilisée. Le tableau II-1-2 montre le développement des variables duales pendant la première partie de l'algorithme, qui calcule la stratégie optimale.

	Etiquettes de noeuds (u_i, f_i)						Lien avec $\text{Min}(u_j+c_a)$			
θ	A	I2	I	Y3	Y	B	(i,j)	f_a	u_j+c_a	arÃ
1	∞,0	∞,0	∞,0	∞,0	∞,0	0,0	(Y3,B)	∞	4.0	yes
2	"	"	"	4.0,∞	"	"	(Y,Y3)	1/15	4.0	yes
3	"	"	"	"	19.0,1/15	"	(I,Y3)	1/15	8.0	yes
4	"	"	23.0,1/15	"	"	"	(Y,B)	1/3	10.0	yes
5	"	"	"	"	11.5,2/5	"	(Y3,Y)	∞	11.5	no
6	"	"	"	"	"	"	(I2,Y)	∞	17.5	yes
7	"	17.5,∞	"	"	"	"	(I,I2)	1/6	17.5	yes
8	"	"	19.0,7/30	"	"	"	(I2,I)	∞	19.0	no
9	"	"	"	"	"	"	(A,I2)	1/6	24.5	yes
10	30.5,1/6	"	"	"	"	"	(A,B)	1/6	25.0	yes
	27.75,1/3	17.5,∞	19.0,7/30	4.0,∞	11.5,2/5	0,0				

TABLEAU II-1-2: Trouver la stratégie optimale.

Le tableau II-1-3 montre les étapes de la deuxième partie de l'algorithme, i.e. le chargement du réseau. On note que la solution optimale trouvée par l'algorithme correspond exactement à la solution optimale qu'on avait déjà rencontrée dans la figure II-1-2.

	Lien		Volumes des noeuds					
θ	(i,j)	Volume	A	X2	X	Y3	Y	B
			1.00	0.00	0.00	0.00	0.00	-1.00
10	(A,B)	0.50	1.00	0.00	0.00	0.00	0.00	-0.50
9	(A,X2)	0.50	1.00	0.50	0.00	0.00	0.00	-0.50
8	(X2,X)	0.00	1.00	0.50	0.00	0.00	0.00	-0.50
7	(X,X2)	0.00	1.00	0.50	0.00	0.00	0.00	-0.50
6	(X2,Y)	0.50	1.00	0.50	0.00	0.00	0.50	-0.50
5	(Y3,Y)	0.00	1.00	0.50	0.00	0.00	0.50	-0.50
4	(Y,B)	0.42	1.00	0.50	0.00	0.00	0.50	-0.08
3	(X,Y3)	0.00	1.00	0.50	0.00	0.00	0.50	-0.08
2	(Y,Y3)	0.08	1.00	0.50	0.00	0.08	0.50	-0.08
1	(Y3,B)	0.08	1.00	0.50	0.00	0.08	0.50	0.00

TABLEAU II-1-3: Affectation de la demande.

Nous allons maintenant examiner la complexité de l'algorithme proposé, en terme de nombre d'opérations nécessaires, i.e. de temps de calcul. Dans le tableau II-1-4, on retrouve les complexités de calculs associées avec chacune des étapes de l'algorithme, ainsi que le nombre de fois que chaque étape est exécutée; m dénote le nombre de liens et n dénote le nombre de noeuds dans le réseau.

Etape	Complexité	#Fois
1.1	O(n+m)	1
1.2	O(m)	m
1.3	O(1)	m
2.1	O(n)	1
2.2	O(m)	1

TABLEAU II-1-4: Complexité des étapes de l'algorithme.

La complexité totale de l'algorithme est déterminée par l'étape 1.2; elle est $O(m^2)$. Notons aussi que dans l'étape 2.2, aucun calcul supplémentaire n'est nécessaire pour établir l'ordre dans lequel les liens sont traités, car il s'agit de l'ordre inverse de celui établi dans la première partie de l'algorithme. Evidemment il est possible d'implanter la recherche du prochain lien à traiter (étape 1.2) à l'aide d'une structure de tas (en anglais: "heap"), et de réduire ainsi la complexité de l'étape 1.2 à $O(\log m)$. La complexité totale de l'algorithme est réduite alors à $O(m \log m)$.

Souvent, le réseau considéré comporte plus d'un noeud de destination. Dans ce cas, l'algorithme est appliqué une fois pour chaque noeud de destination, en accumulant les volumes résultants. Une légère modification de l'algorithme permet dans ce cas d'utiliser les variables v_a pour accumuler les volumes totaux. L'algorithme pour traiter plusieurs destinations est le suivant:

(I) Initialiser les volumes $v_a := 0$, $a \in A$.

(II) Affecter la demande vers chaque destination selon l'algorithme donné ci-haut, en substituant l'étape 2.2 par

2.2' Pour chaque lien $a \in A$, en ordre décroissant de $(u_j + c_a)$:

$$\text{si } a \in \bar{A} \text{ alors } v_a := v_a + \frac{f_a}{f_i} v_i,$$

$$v_j := v_j + \frac{f_a}{f_i} v_i.$$

La complexité de cet algorithme, en supposant que chaque noeud est une destination, est $O(n\ m \log m)$. Puisqu'en plus, $m < n^2$, on voit que la complexité peut aussi être exprimée comme $O(n\ m \log n)$. Si on compare cette complexité avec la complexité d'un algorithme de chemins plus courts, qui est aussi $O(n\ m \log n)$, on s'aperçoit que l'algorithme présenté ici est très efficace et a le potentiel de pouvoir être appliqué même à de très grands réseaux de transport en commun.

1.6 Une extension non-linéaire

Jusqu'à maintenant, on a considéré les temps de déplacement c_a, qui sont associés avec chaque lien $a \in A$, comme des valeurs constantes et connues. On va maintenant étendre le modèle et admettre des fonctions non-linéaires $c_a(v_a)$; le temps de déplacement d'un lien devient une fonction du volume sur ce lien. On va supposer que ces fonctions sont non-négatives, continues et non-décroissantes. Cette relation fonctionnelle entre le temps de déplacement et le volume du lien peut être utilisée pour modéliser le changement de la performance d'un véhicule sous différentes conditions de chargement, par exemple en montant une pente. Aussi, mais d'une façon limitée, $c_a(v_a)$ peut représenter des effets de congestion qui sont dûs à la capacité finie des véhicules. Plus les véhicules deviennent chargés, ou même surchargés, plus l'inconfort des passagers augmente et moins ils considèrent ce lien (ou cette ligne) comme étant attrayant. Dans ce cas, $c_a(v_a)$ n'est plus un temps de déplacement pur, mais il inclut une pondération variable de ce temps, selon le degré de congestion dans les véhicules. Cette approche permet de modéliser certaines des conséquences de la capacité finie des véhicules de transport en commun; les fonctions

$c_a(v_a)$ ressemblent aux fonctions volume-délai qui sont utilisées dans les modèles d'affectation routière pour représenter la capacité finie des rues. Il faut quand même noter que cette approche induit les limitations suivantes:

- Dans ce modèle, tous les passagers à bord d'un véhicule sont sujets au même degré d'inconfort. On ne distingue pas les passagers qui sont déjà dans le véhicule (qui ont une plus grande probabilité d'avoir une place assise) des passagers qui viennent de monter à bord.

- Le temps d'attente ne dépend pas directement des volumes de passagers déjà à bord des véhicules. Les passagers ne doivent jamais attendre l'arrivée du deuxième véhicule de la même ligne; les files d'attente aux arrêts ne sont pas considérées dans le modèle.

- Le temps d'échange de passagers qu'un véhicule passe à un arrêt ne dépend pas du nombre de personnes qui montent et qui descendent.

Malgré ces limitations, qui deviennent importantes surtout pour des réseaux fortement congestionnés, nous considérons qu'il est important de poursuivre cette extension du modèle. Pour des situations de faible et moyenne congestion, le modèle représente bien les conséquence de la congestion. C'est seulement pour les situations de forte congestion, où une partie importante des passagers doit attendre l'arrivée du deuxième ou troisième véhicule de la même ligne avant qu'il puisse monter à bord, que les limitations deviennent non-négligeables. Le modèle qu'on présentera ci-bas est à notre connaissance le seul qui inclut des effets de congestion dans un modèle d'affectation pour le transport en commun, d'une façon non-

heuristique, tout en restant facile à résoudre même pour des réseaux de grande taille.

Nous dénoterons par v_a^r la partie du volume sur le lien $a \in A$ qui est destinée vers le noeud destination $r \in R$. Le volume total sur un lien a, v_a, est donc donné par

$$v_a = \sum_{r \in R} v_a^r , \qquad a \in A. \tag{43}$$

A cause du fait que le temps de déplacement est une fonction du volume total sur le lien, il n'est plus possible de séparer le problème par destination.

Soit g_i^r la demande du noeud $i \in I$ à la destination $r \in R$. Soit en plus K_r l'ensemble de toutes les stratégies réalisables pour atteindre la destination r. On dénotera par h_i^k la partie de la demande g_i^r qui est affectée selon la stratégie $k \in K_r$. Evidemment, on exige la conservation du flôt, ce qui nous donne

$$\sum_{k \in K_r} h_i^k = g_i^r , \qquad i \in I,\ k \in K_r,\ r \in R. \tag{44}$$

Comme plus haut, nous supposerons encore que le voyageur se déplace selon la stratégie qui minimise l'espérance de son temps de voyage. Mais maintenant, le temps de voyage pour une stratégie dépend des volumes totaux sur les liens, et par conséquent, des stratégies choisies par les autres voyageurs, qui, eux aussi, essaient de minimiser leur temps de voyage. Mathématiquement, on peut caractériser le comportement des voyageurs par des conditions d'équilibre, similaires à celles qu'on dérive du deuxième principe de Wardrop pour caractériser le comportement des automobilistes

(Wardrop, 1952). Soit s_i^k l'espérance du temps de voyage de i à r en utilisant la stratégie $k \in K_r$, et soit u_i^r le minimum des espérances du temps de voyage de i à r. Les conditions d'équilibre peuvent s'exprimer alors comme

$$\begin{aligned} s_i^{k^*} &= u_i^{r^*}, \text{ si } h_i^{k^*} > 0 \\ &, \qquad i \in I,\ k \in K_r,\ r \in R, \qquad (45) \\ s_i^{k^*} &\geq u_i^{r^*}, \text{ si } h_i^{k^*} = 0 \end{aligned}$$

où l'étoile (*) est utilisée pour dénoter une solution d'équilibre. Les conditions (45) signifient que seulement les stratégies à temps minimum sont utilisées ($h_i^k > 0$), tandis que les stratégies à temps non-minimal ne sont pas utilisées ($h_i^k = 0$).

Proposition 8: Chaque solution optimale du problème suivant P8 satisfait les conditions d'équilibre (45).

$$\text{Min} \sum_{a \in A} \int_0^{v_a} c_a(x)dx + \sum_{i \in I} \sum_{r \in R} w_i^r \qquad (P8)$$

sujet à (43) et

$$v_a^r \leq f_a w_i^r, \qquad a \in A_i^+,\ i \in I,\ r \in R, \qquad (46)$$

$$\sum_{a \in A_i^+} v_a^r = \sum_{a \in A_i^-} v_a^r + g_i^r, \qquad i \in I,\ r \in R, \qquad (47)$$

$$v_a^r \geq 0, \qquad a \in A,\ r \in R. \qquad (48)$$

Preuve: Notons d'abord que les contraintes du problème P8 sont séparables par destination r. Les points extrémaux de la région réalisable associée avec la destination r correspondent exactement à l'ensemble des stratégies réalisables K_r pour atteindre la destination r, comme on l'a démontré dans

la proposition 4. Il est donc possible de reformuler le problème P8 en termes des variables v_a^k, qui représentent la partie du volume total du lien a qui est affectée selon la stratégie $k \in K_r$.

$$\text{Min} \sum_{a \in A} \int_0^{v_a} c_a(x)dx + \sum_{r \in R} \sum_{k \in K_r} \sum_{i \in I} w_i^k \qquad \text{(P9)}$$

sujet à (44) et

$$v_a^k = f_a \delta_a^k w_i^k , \qquad a \in A_i^+,\ i \in I,\ k \in K_r,\ r \in R, \qquad (49)$$

$$\sum_{a \in A_i^+} v_a^k = \sum_{a \in A_i^-} v_a^k + h_i^k , \qquad i \in I,\ k \in K_r,\ r \in R, \qquad (50)$$

$$v_a = \sum_{i \in I} \sum_{k \in K_r} v_a^k , \qquad a \in A, \qquad (51)$$

$$w_i^k \geq 0 ,\ h_i^k \geq 0 , \qquad i \in I,\ k \in K_r,\ r \in R. \qquad (52)$$

Les constantes δ_a^k sont 1 si le lien a appartient à la stratégie k et 0 sinon. Soient $\hat{\mu}_a^k$, $\hat{s}_i^k$ et $\hat{u}_i^r$ les variables duales qui correspondent aux contraintes (49), (50) et (44) à l'optimum. Les conditions d'optimalité de Kuhn-Tucker, qui sont nécessaires et suffisantes pour caractériser une solution optimale (v^*, w^*, h^*) du problème P9, peuvent s'écrire comme

$$\hat{s}_j^k + c_a(v_a^*) + \hat{\mu}_a^k \begin{cases} = \hat{s}_i^k, \text{ si } h_i^{k*} > 0 \\ \geq \hat{s}_i^k, \text{ si } h_i^{k*} = 0 \end{cases} \qquad a=(i,j) \in A, k \in K_r, r \in R, \qquad (53)$$

$$\sum_{a \in A_i^+} \hat{\mu}_a^k \, \delta_a^k \, f_a = 1 \,, \qquad i \in I,\ k \in K_r,\ r \in R \qquad (54)$$

et

$$\begin{aligned} \hat{s}_i^k &= \hat{u}_i^r \quad ,\text{si } h_i^{k^*} > 0 \\ \hat{s}_i^k &\geq \hat{u}_i^r \quad ,\text{si } h_i^{k^*} = 0 \end{aligned} \,, \qquad i \in I,\ k \in K_r,\ r \in R. \qquad (55)$$

Dans (53) et (54), nous reconnaissons la même structure que celle de (24) et (25). Les variables $\hat{s}_i^k$ correspondent donc à l'espérance du temps de voyage du noeud i à la destination r en utilisant la stratégie k; i.e. on a

$$\hat{s}_i^k = s_i^{k^*} \,, \qquad i \in I,\ k \in K_r,\ r \in R. \qquad (56)$$

Mais puisque (55) implique que $\hat{u}_i^r = \min_{k \in K_r} \hat{s}_i^k$, on a

$$\hat{u}_i^r = u_i^{r^*} \,, \qquad i \in I,\ r \in R \qquad (57)$$

et cela prouve que (55) est équivalent à (45). ■

Comme le problème P8 est un problème de minimisation convexe avec des contraintes linéaires, il peut être résolu par n'importe quelle des méthodes standard applicables à cette classe de problèmes. On discute brièvement un algorithme de résolution qui est basé sur la méthode des approximations linéaires successives (Frank et Wolfe, 1956). Cet algorithme n'a pas besoin de stocker explicitement les volumes de lien par destination, v_a^k, mais uniquement les volumes totaux. Il est donc possible de l'appliquer à des réseaux de grande taille. L'approximation linéaire de la fonction objective du problème P8 résulte dans un problème qui est

séparable par destination. Pour chaque destination le sous-problème ainsi obtenu est équivalent au problème d'affectation pour le transport en commun sans congestion, qu'on a discuté dans la section 1.4, et qui peut être résolu efficacement par l'algorithme donné dans la section 1.5. L'algorithme d'affectation pour le transport en commun avec des fonctions de coût non-linéaire est alors:

0. (Initialisation) Trouver une solution réalisable du problème P8, soit (v^0,w^0). Mettre x := 0.

1. (Sous-problème linéaire)

Mettre x := x + 1.

Obtenir $(\hat{v},\hat{w})$ comme solution de l'algorithme de la section 1.5 avec des temps de déplacement $c_a=c_a(v_a^{x-1})$ pour chacune des destinations $r\in R$.

2. (Recherche unidimensionnelle)

Trouver λ^x qui minimise la fonction objective de P8 sur le segment

$(1-\lambda)(v^{x-1},w^{x-1})+\lambda(\hat{v},\hat{w})$, $0 \leq \lambda \leq 1$.

3. (Mise à jour) $(v^x,w^x) := (1-\lambda^x)(v^{x-1},w^{x-1})+\lambda^x(\hat{v},\hat{w})$.

Si $\|(v^x,w^x)-(v^{x-1},w^{x-1})\| < \varepsilon$ alors ARRETER

sinon aller à l'étape 1.

1.7 Conclusions

Dans les sections précédentes, nous avons développé une nouvelle approche pour formuler et résoudre le problème de l'affectation du transport en commun. Cette approche se distingue en deux aspects importants de toutes les méthodes existantes.

Premièrement, toutes les méthodes existantes limitent le choix de route d'un passager individuel au choix d'un chemin sur le réseau, que ce soit un chemin en terme de segments de ligne, liens du réseau ou noeuds de transfert. Dans notre modèle, par contre, le voyageur choisit une stratégie qui contient un ensemble de chemins attrayants. Face à ces chemins-là, le passager est indifférent, i.e. il va prendre la première occasion qui s'offre. Le trajet emprunté pour un voyage particulier dépend donc de l'ordre (aléatoire) dans lequel les véhicules des diverses lignes arrivent à l'arrêt. Cette description de l'ensemble de choix du voyageur permet de modéliser de façon plus réaliste le comportement des passagers.

Deuxièmement, notre approche permet, pour la première fois, de formuler le problème de choix de route dans un réseau de transport en commun comme un problème de programmation mathématique basé sur des hypothèses de comportement réalistes et pouvant être résolu exactement, sans recours à des méthodes heuristiques.

Dans sa forme générale, le modèle permet des distributions statistiques quelconques pour l'intervalle entre l'arrivée de deux véhicules de la même ligne. En se limitant au cas spécial des fréquences combinées, ce qui correspond à des distributions exponentielles, le modèle peut être

formulé comme un problème de programmation linéaire, qui a des similarités avec le modèle d'affectation "tout-ou-rien". Nous avons développé un algorithme de résolution qui est de complexité polynomiale. Cet algorithme n'est que légèrement plus compliqué que l'algorithme de plus courts chemins de Dijkstra; il est donc applicable même à des réseau de transport en commun de très grande taille.

Comme extension du modèle, nous avons montré qu'il peut facilement être généralisé pour permettre des coûts de lien dépendant monotonement du volume sur le lien. Dans ce cas, la solution optimale peut être caractérisée par des conditions d'équilibre similaires à celles du modèle d'affectation routière. Nous avons démontré qu'il existe une formulation équivalente de programmation convexe, et nous avons proposé un algorithme efficace, basé sur la méthode de Frank et Wolfe.

Considérant la simplicité stupéfiante du modèle et de l'algorithme, on peut être surpris que cette approche n'ait pas été découverte plus tôt. Deux raisons historiques semblent expliquer, au moins en partie, pourquoi cette approche n'a pas été explorée pendant si longtemps:

- Les algorithmes de plus courts chemins étaient connus depuis longtemps. Tous les modèles d'affectation routière sont basés sur le choix d'un chemin et, par conséquent, incorporent des algorithmes de plus courts chemins. Vu de façon historique, il n'est donc pas surprenant que, confronté au problème d'affectation du transport en commun, on a pris pour acquis que le choix de route d'un passager de transport en commun est, lui aussi, basé sur le choix d'un chemin.

- On est habitué d'appliquer les algorithmes de plus courts chemins, ou d'autres techniques d'affectation, à une origine à la fois, pour calculer simultanément les chemins de cette origine à toutes les destinations. Le problème de plus court chemin, tout comme celui de l'affectation routière en général, est, en soi, complètement symétrique par rapport aux origines et aux destinations. Le choix de travailler à partir des origines est donc purement arbitraire, i.e. on pourrait aussi bien calculer simultanément les chemins de toutes les origines à une destination. Dans le cas du réseau de transport en commun, cette symétrie n'existe plus; elle est détruite par le phénomène de l'attente, qui est toujours localisé au point d'embarquement. C'est pour cette raison que la stratégie optimale ne peut être calculée efficacement qu'à partir de la destination - ce qui, malheureusement, ne correspond pas à la façon habituelle d'approcher un problème.

L'approche décrite ici a été programmée et intégrée dans le cadre du système EMME/2, qui fait l'objet du chapitre III. Dans la section III.5.5 nous verrons comment cet algorithme a été adapté à la structure d'un réseau de transport en commun réaliste.

Il y a deux types de généralisation du modèle qui pourraient servir à mieux modéliser les effets de la congestion. Les deux pourraient constituer des points de départ pour des recherches futures intéressantes. Premièrement, il est relativement facile d'étendre le modèle pour permettre des coûts non-diagonaux asymétriques sur les liens. Cette généralisation nous amène à une formulation d'inégalité variationnelle. Tous les

résultats obtenus pour le problème d'affectation d'équilibre avec coûts asymétriques peuvent être généralisés à ce modèle (Smith, 1979; Aashtiani et Magnanti, 1980; Dafermos, 1980; Fisk et Nguyen, 1980; Florian et Spiess, 1981). Malheureusement, ces résultats supposent la monotonicité de la fonction de coût - une hypothèse qui ne tient pas pour les effets de congestion qu'on désire modéliser. La deuxième façon de généraliser le modèle utilise les fréquences pour modéliser la congestion. Ceci est très réaliste, puisque, dans un réseau de transport en commun, la congestion affecte plus le temps d'attente que le temps de déplacement. Une fonction $f_1(v)$ pourrait représenter la fréquence effective, telle que perçue par le passager. Puisque ce type de généralisation affecte la répartition des voyages à l'intérieur d'une stratégie, elle n'est pas triviale du tout - un défi pour des recherches futures!

2. Un modèle de vraisemblance maximale pour l'estimation d'une matrice origine - destination

2.1 Introduction

Le problème de reconstituer une matrice origine - destination à partir des volumes observés sur les liens du réseau a été étudié par plusieurs auteurs (voir par exemple Jörnsten and Nguyen (1979), Van Zuylen et Willumsen (1980) et Van Zuylen (1981) pour les références les plus récentes). Nguyen (1982) donne un bon aperçu de l'état des connaissances dans ce domaine. Le problème général peut être formulé comme suit. Soit P l'ensemble des sommets-origines, Q l'ensemble des sommets-destinations et I=PxQ l'ensemble des paires origine - destination (ou en plus bref, paires O-D). La plupart des modèles existants pour estimer une matrice O-D $(T_i, i\in I)$ peuvent être formulés comme

$$\text{Min} \quad f(T,t) \tag{1}$$

sujet à

$$\sum_{i\in I} n_{ia} T_i = V_a \qquad ,a\in A, \tag{2}$$

$$T_i \geq 0 \qquad ,i\in I. \tag{3}$$

où les coefficients n_{ia} correspondent à la proportion des voyages de la paire O-D i qui traversent le lien a, et V_a dénote le volume observé sur le lien a. La matrice $(t_i, i\in I)$ contient de l'information à priori sur la matrice (T_i). La fonction objective f(T,t) est, ou bien une mesure de dis-

tance entre $\{T_i\}$ et $\{t_i\}$ en utilisant une métrique particulière (Deming, 1940; Smith et Hutchinson, 1981), ou bien la fonction convexe d'entropie, utilisée par plusieurs auteurs (Jörnsten and Nguyen, 1979; Nguyen, 1982; Smith et Hutchinson, 1981; Van Vliet et Willumsen, 1981; Van Zuylen et Willumsen, 1980).

$$f(T,t) = \sum_{i \in I} T_i \log \frac{T_i}{t_i}. \qquad (4)$$

Bien que ces formes particulières de la fonction f(T,t) sont très plausibles, le choix de la fonction qu'il faut utiliser pour une application particulière est plutôt arbitraire. D'un côté, aucune des méthodes qu'on retrouve dans la littérature n'associe une interprétation précise à la matrice $\{t_i\}$. On ne trouve que de vagues descriptions pour $\{t_i\}$, comme par exemple "estimation initiale", "vieille matrice O-D", "a priori guess", "matrice visée" et "matrice de référence" (Jörnsten and Nguyen, 1979; Nguyen, 1982; Van Vliet et Willumsen, 1981; Van Zuylen, 1981). D'un autre coté, il est bien établi que "la qualité des estimations résultantes est extrêmement sensible à l'information a priori contenue dans la matrice de référence $\{t_i\}$" (Nguyen, 1982). En regardant ce rôle de $\{t_i\}$ qui est plutôt imprécis, pas encore compris à fond et peut-être même confus, ce n'est pas surprenant qu'il manque à tous les modèles développés jusqu'à date, la possibilité d'inférence statistique sur la validité des estimations résultantes.

Le but de cette recherche est de montrer qu'il est possible de surmonter les difficultés mentionnées ci-haut en spécifiant une interpretation précise pour $\{t_i\}$. En réduisant ainsi la classe de problèmes considérés, il devient possible de gagner une compréhension approfondie de la structure

de ces problèmes. Dans ce qui suit, nous considérons le cas pour lequel la matrice $\{t_i\}$ a été obtenue en observant pour chaque paire O-D i une variable aléatoire distribuée selon une loi de Poisson avec moyenne inconnue T_i. Ainsi, $\{t_i\}$ est une observation d'un ensemble de variables aléatoires X_i, $i \in I$, distribuées selon des distributions de Poisson indépendantes, avec moyenne T_i. C'est d'ailleurs la même hypothèse sur la matrice $\{t_i\}$ que celle utilisée par Kirby (1974) pour l'étude des problèmes de calibration des modèles de gravité. Cette hypothèse, faite sur la provenance de la matrice $\{t_i\}$, correspond bien à beaucoup de situations qu'on retrouve en pratique. La plupart des techniques utilisées pour faire des enquêtes origine - destination sont basées sur un comptage de voyages individuels, et produisent, par conséquent, des matrices observées à valeurs entières. Quand le pourcentage des voyages observés devient petit, on sait que la distribution multinominale converge asymptotiquement vers des distributions de Poisson indépendantes. La plupart des enquêtes sur les voyages ayant pour but le travail appartiennent à cette catégorie. Une autre catégorie d'enquêtes qui a comme résultat des matrices observées $\{t_i\}$, distribuées selon des lois de Poisson, correspond au cas où on observe des événements mutuellement indépendants pour une période de temps finie. Des voyages hors de l'heure de pointe et des appels téléphoniques se placent bien dans cette catégorie.

Le but de notre modèle est d'estimer les paramètres $\{T_i\}$ des distributions de Poisson en utilisant la matrice observée $\{t_i\}$ ainsi que les volumes observés sur les liens du réseau. Puisque les volumes des liens sont observés souvent sur une base journalière, nous pouvons supposer connaître leurs "vraies" valeurs moyennes ou au moins une très bonne approximation.

Dans les sections suivantes, nous ne considérons que le cas où les proportions d'affectation n_{ia} sont des constantes connues et déterminées d'avance. Sans manquer de généralité, nous supposons aussi que l'ensemble réalisable défini par (2) et (3) ne restreint aucune des variables T_i à une seule valeur. Si jamais ce cas se présentait, on pourrait simplement éliminer cette variable. Ceci aurait comme une conséquence que $V_a>0$, $a\in A$, puisque $V_a=0$ forcerait $T_i=0$ pour tout $i\in I$ avec $n_{ia}>0$.

Ce chapitre est organisé de la façon suivante: dans la section 2.2, nous élaborons le modèle de vraisemblance maximale et nous étudions ses propriétés mathématiques. Dans la section 2.3, un algorithme de résolution est proposé qui est basé sur la méthode de descente cyclique au long des axes de coordonnées, dont les propriétes de convergence sont analysées. La section 2.4 traite le cas spécial où, au lieu des volumes sur les liens, les totaux marginaux de la matrice O-D (productions et attractions) sont connus. La section 2.5 contient des résultats permettant l'inférence statistique sur la validité du modèle et dans la section 2.6, nous élaborons une extension du modèle pour traiter le cas où, au lieu de connaître les vraies valeurs des volumes sur les liens, on ne dispose que des observations provenant des distributions de Poisson. La section 2.7 contient les conclusions.

2.2 Un modèle de vraisemblance maximale

Selon l'hypothèse que t_i est obtenu en observant, pour chaque paire O-D, une variable aléatoire distribuée selon une loi de Poisson avec moyenne T_i, la probabilité d'observer t_i est

$$\text{Prob}\{\ \text{Poisson}(T_i)=t_i\} = \frac{T_i^{t_i} e^{-T_i}}{t_i!}, \quad i\in I. \tag{5}$$

La probabilité conjointe d'observer la matrice (t_i) est alors

$$\text{Prob}\{\ (t_i)\ \} = \prod_{i\in I} \text{Prob}[t_i] = \prod_{i\in I} \frac{T_i^{t_i} e^{-T_i}}{t_i!}. \tag{6}$$

Puisqu'en plus, nous savons que la matrice (T_i) satisfait (2) et (3), elle est non-négative et consistante avec les volumes sur les liens V_a, $a\in A$.

Nous appliquons la méthode de vraisemblance maximale (Wilks, 1962; Wilson, 1976) pour trouver la matrice (T_i^*) qui satisfait les contraintes (2) et (3) en maximisant la probabilité (6) d'observer (t_i). En remplaçant la fonction de vraisemblance (6) par son logarithme, et en adoptant la convention $0 log 0 = 0$, nous pouvons formuler le modèle de vraisemblance maximale comme

$$\text{Max} \sum_{i\in I} (t_i log T_i - T_i - log t_i!) \tag{7}$$

sujet à (2) et (3).

Puisque le dernier terme dans (7) est une constante, la fonction objective (7) est équivalente à

$$\text{Min} \sum_{i \in I} (T_i - t_i \log T_i) \qquad (8)$$

sujet à (2) et (3). Nous avons donc trouvé un autre problème de la famille des problèmes (1) avec

$$f(T,t) = \sum_{i \in I} (T_i - t_i \log T_i) \qquad (9)$$

Notons que (8) est une fonction convexe en T et, de plus, strictement convexe en T lorsque $t_i > 0$, $i \in I$. Si les volumes V_a, $a \in A$, sont réalisables et l'ensemble des contraintes (2) et (3) est consistant, alors l'existence d'une solution optimale de (8) est assurée puisque le problème non-contraint admet une solution optimale finie $\{T_i\}=\{t_i\}$. Une solution optimale du probème (8) peut être obtenue en appliquant une des méthodes standard pour la résolution des problèmes de programmation convexe. Dans la section 3, un algorithme qui est adapté spécialement pour ce problème, sera développé .

Soit T^* une solution optimale du problème (8). Comme nous verrons dans le développement suivant, il est important de classifier les paires O-D dans deux groupes, selon que l'élément correspondant de la matrice observée t_i soit vide ou non. Ainsi, nous dénotons par I^+ l'ensemble des paires O-D pour lesquelles $t_i > 0$ et I^o, l'ensemble des paires O-D pour lesquelles $t_i = 0$.

Proposition 1: T_i^*, $i \in I^+$, sont uniques.

Preuves: Le problème (8) est convexe et la fonction objective est séparable par paire O-D. La partie de (8) qui correspond à $i \in I^+$ est strictement convexe et il s'en suit que les T_i^*, $i \in I^+$, sont uniques.■

Proposition 2: Nous supposons connues les valeurs optimales T_i^*, $i\varepsilon I^+$. Alors, les valeurs optimales T_i^*, $i\varepsilon I^o$ correspondent à la solution optimale du problème de programmation linéaire suivant

$$\text{Min} \sum_{i\varepsilon I^o} T_i \tag{10}$$

sujet à

$$\sum_{i\varepsilon I^o} n_{ia}T_i = V_a - \sum_{i\varepsilon I^+} n_{ia}T_i^* , \qquad a\varepsilon A, \tag{11}$$

$$T_i \geq 0 , \qquad i\varepsilon I^o. \tag{12}$$

Preuve: Séparons d'abord les paires O-D dans (8) selon leur appartenance à I^o et I^+

$$\text{Min} \sum_{i\varepsilon I^o} T_i + \sum_{i\varepsilon I^+} (T_i - t_i log T_i) \tag{13}$$

sujet à

$$\sum_{i\varepsilon I^o} n_{ia}T_i + \sum_{i\varepsilon I^+} n_{ia}T_i = V_a , \qquad a\varepsilon A \tag{14}$$

et (3). Puisque nous connaissons les valeurs optimales de T_i, $i\varepsilon I^+$, nous pouvons les substituer dans (13) et (14). Le deuxième terme dans (13) est maintenant constant et il peut être éliminé, résultant en (11).■

Proposition 3: Le nombre total de voyages $\sum_{i\varepsilon I} T_i^*$ d'une solution optimale de (8) est unique.

Preuve: Selon la proposition 1, les valeurs de T_i^*, $i\varepsilon I^+$, sont uniques et alors $\sum_{i\varepsilon I^+} T_i^*$ est aussi unique. En utilisant la proposition 2, nous voyons

que $\sum_{i\in I^o} T_i^*$ est la valeur de la fonction objective du problème (11) à l'optimum et est donc nécessairement unique. Puisque

$$\sum_{i\in I} T_i^* = \sum_{i\in I^o} T_i^* + \sum_{i\in I^+} T_i^*,$$

le nombre total de voyages d'une solution optimale de (8) est unique.■

L'unicité du nombre total de voyages est une propriété importante du modèle proposé, parce que dans le cas où $t_i=0$ a été observé pour certaines paires O-D, la matrice T^* n'est pas nécessairement unique. Si par contre le nombre total de voyages est connu grâce à d'autres sources de données, on peut facilement en tenir compte en ajoutant une contrainte supplémentaire à (2), i.e. en posant $n_{ia}=1$, $i\in I$ et V_a égal au nombre total de voyages.

Nous poursuivons notre analyse en étudiant la formulation duale du problème (8). Soit λ_a, $a\in A$, les multiplicateurs de Lagrange associés aux contraintes (2). Le dual Lagrangéen de (8) est alors

$$\underset{\lambda}{\text{Max}}\left\{ \underset{T\geq 0}{\text{Inf}}\left(\sum_{i\in I} (T_i - t_i \log T_i) + \sum_{a\in A} \lambda_a \left(\sum_{i\in I} n_{ia} T_i - V_a \right) \right)\right\} \tag{15}$$

En réarrangeant les termes et en changeant l'ordre de sommation, on obtient

$$\underset{\lambda}{\text{Max}}\left\{ \underset{T\geq 0}{\text{Inf}}\left(\sum_{i\in I} T_i \left(1+ \sum_{a\in A} n_{ia}\lambda_a\right) - \sum_{i\in I} t_i \log T_i - \sum_{a\in A} \lambda_a V_a \right) \right\} \tag{16}$$

et en notant que les sous-problèmes de trouver l'infimum sont séparables par paire O-D, (16) devient

$$\underset{\lambda}{\text{Max}}\left\{ \sum_{i\in I} \underset{T_i\geq 0}{\text{Inf}} \left(T_i \left(1+ \sum_{a\in A} n_{ia}\lambda_a\right) - t_i \log T_i\right) - \sum_{a\in A} \lambda_a V_a \right\} . \tag{17}$$

Cette dernière formulation montre d'une façon explicite les conditions que λ_a doit satisfaire pour constituer un point essentiellement réalisable; i.e., l'infimum de (16) n'admet une valeur finie que pour les multiplicateurs λ_a, $a\varepsilon A$, qui satisfont

$$1 + \sum_{a\varepsilon A} n_{ia}\lambda_a \begin{cases} > 0 & , s_i\ t_i > 0, \\ \\ \geq 0 & , s_i\ t_i = 0, \end{cases} \qquad i\varepsilon I. \tag{18}$$

Pour les multiplicateurs qui satisfont (18), il est facile d'évaluer T_i auquel l'infimum est atteint

$$\hat{T}_i(\lambda) = \begin{cases} \dfrac{t_i}{1 + \sum_{a\varepsilon A} n_{ia}\lambda_a} & , s_i\ t_i > 0, \\ \\ 0 & , s_i\ t_i = 0, \end{cases} \qquad i\varepsilon I. \tag{19}$$

En substituant (19) dans (16) et en négligeant des termes constants, on obtient une nouvelle formulation du dual qui ne dépend plus explicitement des variables primales T_i, mais seulement des variables duales λ_a:

$$\text{Min } g(\lambda) = \sum_{i\varepsilon I^+} t_i \log \frac{t_i}{1 + \sum_{a\varepsilon A} n_{ia}\lambda_a} + \sum_{a\varepsilon A} \lambda_a V_a \tag{20}$$

sujet à (18).

Soit T^* une solution optimale du problème (8) et λ^* une solution optimale du problème dual (20). (T^*, λ^*) doivent alors satisfaire aux conditions d'optimalité de Kuhn-Tucker

$$1 + \sum_{a\varepsilon A} n_{ia}\lambda_a^* \begin{cases} = \dfrac{t_i}{T_i^*} & , \text{ si } T_i^* > 0, \\ \geq 0 & , \text{ si } T_i^* = 0, \end{cases} \qquad i\varepsilon I. \tag{21}$$

Notons que, puisque le problème considéré est convexe, (21) est une condition nécessaire et suffisante pour l'optimalité de (T^*,λ^*).

Proposition 4: Les conditions d'optimalité (21) sont équivalentes à

$$T_i^* \left(1 + \sum_{a\varepsilon A} n_{ia}\lambda_a^*\right) = t_i , \qquad i\varepsilon I. \tag{22}$$

Preuve: Nous démontrons d'abord que (21) implique (22). Pour les paires O-D $i\varepsilon I^O$, la condition (21) est simplement une condition de complémentarité et (22) peut être obtenu par la multiplication des termes correspondants. Pour $i\varepsilon I^+$, nous notons d'abord que nécessairement $T_i^* > 0$, car sinon, la valeur de la fonction objective (8) devient infinie, ce qui serait une contradiction de l'optimalité de T^*. Conséquemment, seulement la première partie de (21) s'applique à $i\varepsilon I^+$ et cela est évidemment équivalent à (22).

Nous démontrons maintenant que (22) implique (21). Pour $T_i^* > 0$, (21) est obtenu de (22) en divisant par T_i^*. Pour $i\varepsilon I^O$, (21) est toujours satisfait puisque λ^* est dual-réalisable.■

2.3 Un algorithme de résolution

Les problèmes d'estimation de matrice qu'on rencontre en pratique sont typiquement d'une très grande taille (des centaines de liens et des milliers de paires O-D). Des algorithmes qui ont besoin d'avoir accès à toutes les données du problème en même temps peuvent se révéler impossibles à implanter sur la plupart des ordinateurs normalement disponibles, à cause des limitations de la mémoire. C'est pour cela qu'on discutera dans cette section d'un algorithme très simple, basé sur l'approche de relaxation successive de toutes les contraintes (2) sauf une. De cette façon, on arrive à morceler le problème en sous-problèmes plus petits, dont chacun ne dépend que des données associées avec un seul lien, résultant dans une méthode qu'on peut appliquer efficacement à des problèmes de très grande taille. L'idée principale de cette approche est similaire à celle utilisée pour les méthodes de balancement qui résolvent le modèle de maximisation d'entropie (4), comme par exemple la méthode proposée par Murchland (1977). Mais contrairement au modèle de maximisation d'entropie (4), on ne peut pas adapter la preuve de convergence de Bregman (Bregman, 1967; Lamond et Stewart, 1981) pour notre modèle de vraisemblance maximale (8).

Dans un premier temps, nous faisons l'hypothèse que $t_i > 0$ est vraie pour toutes les paires O-D $i \in I$ et nous examinons l'algorithme suivant:

Algorithme 1:

ETAPE 0: Mettre $\lambda_a := 0$, $a \in A$.

ETAPE 1: Pour chaque $a \in A$ faire:

a) Calculer

$$\lambda_a^{min} = - \min_{\substack{i \in I \\ n_{ia} > 0}} \frac{1 + \sum_{b \neq a} n_{ib} \lambda_b}{n_{ia}} \tag{23}$$

b) Trouver $\bar{\lambda}_a \in (\lambda_a^{min}, \infty)$ qui satisfait

$$\sum_{i \in I} \frac{n_{ia} t_i}{1 + \sum_{b \neq a} n_{ib} \lambda_b + n_{ia} \bar{\lambda}_a} = V_a. \tag{24}$$

c) Mettre $d_a := \bar{\lambda}_a - \lambda_a$ et $\lambda_a := \bar{\lambda}_a$.

ETAPE 2: Si $\|d\| > \varepsilon$ alors retourner à l'ETAPE 1, autrement calculer

$$T_i^* = \frac{t_i}{1 + \sum_{a \in A} n_{ia} \lambda_a}, \qquad i \in I. \tag{25}$$

et **ARRETER.**

Proposition 5: Pour $t_i > 0$, $i \in I$, l'algorithme 1 converge vers la solution optimale du problème (8).

Preuve: Il suffit de démontrer que les valeurs des variables duales λ_a, $a \in A$ convergent vers une solution optimale du problème dual (20). Puisque la matrice (t_i) est positive, (18) ne contient que des inégalités strictes. La solution optimale de (20) doit donc être un point situé à l'intérieur de la région réalisable, et par conséquent on peut appliquer des techniques

d'optimisation sans contraintes. L'étape 1 de l'algorithme correspond à une itération de la méthode de descente cyclique au long des axes de coordonnées, appliquée au problème (20). Le sous-problème uni-dimensionnel qu'on obtient de (20) en gardant constantes toutes les valeurs des variables duales, sauf λ_a, devient

$$\text{Min } g(\lambda_a) = \sum_{i \in I} t_i \log \frac{t_i}{1 + \sum_{b \neq a} n_{ib}\lambda_b + n_{ia}\lambda_a} + \sum_{b \neq a} \lambda_b V_b + \lambda_a V_a. \qquad (26)$$

Dans l'étape 1a) une borne pour λ_a est calculée pour s'assurer que λ reste réalisable. Notons que $\frac{dg(\lambda_a)}{d\lambda_a}$ est une fonction rationnelle avec des pôles à $\lambda_a = -\frac{1 + \sum_{b \neq a} n_{ib}\lambda_b}{n_{ia}}$ pour tous les $a \in A$ avec $n_{ia} > 0$. Puisque nous obligeons λ_a à prendre une valeur dans l'intervalle $(\lambda_a^{min}, \infty)$, nous sommes sûrs que dans (24) $\frac{dg(\lambda_a)}{d\lambda_a} = 0$ a exactement une solution. Puisque la méthode de descente cyclique au long des axes de coordonnées est globalement convergente pour les problèmes de programmation convexe sans contraintes, les valeurs de λ_a, $a \in A$, convergent vers une solution optimale de (20). Les valeurs optimales des variables primales peuvent alors être obtenues en utilisant les conditions d'optimalité (22), ce qui est fait dans l'étape 2 de l'algorithme. ■

Bien que la convergence de l'algorithme 1 ait été prouvée seulement pour le cas $t_i > 0$, $i \in I$, il est quand même possible de l'appliquer dans des cas où la demande observée pour certaines paires O-D est zéro. Cela peut être fait en appliquant l'algorithme 1 à la matrice (t_i'), qui est obtenue

de $\{t_i\}$ en remplaçant tous les zéros par ε, un très petit nombre positif.

On peut aussi appliquer directement la méthode de descente cyclique au long des axes de coordonnées dans le cas où I^o n'est pas vide. Cela donne lieu à l'algorithme suivant:

Algorithme 2:

ETAPE 0: Mettre $\lambda_a := 0$, $a \in A$; mettre $s_i := 1$, $i \in I$.

ETAPE 1: Pour chaque $a \in A$ faire:

a) Calculer $d_a^{min} = - \underset{\substack{i \in I \\ n_{ia} > 0}}{\text{Min}} \frac{s_i}{n_{ia}}$ et soit $\bar{I}$ l'ensemble de paires O-D pour lesquelles le minimum est atteint.

b) Si $\bar{I} \subseteq I^o$ et $\sum_{i \in I^+} \frac{n_{ia} t_i}{s_i + n_{ia} d_a^{min}} \leq V_a$

alors mettre $d_a = d_a^{min}$,

autrement trouver $d_a > d_a^{min}$ qui satisfait

$$\sum_{i \in I^+} \frac{n_{ia} t_i}{s_i + n_{ia} d_a} = V_a .$$

c) Mettre $\lambda_a := \lambda_a + d_a$ et $s_i := s_i + n_{ia} d_a$, $i \in I$.

ETAPE 2: Si $\|d\| > \varepsilon$ alors retourner à l'ETAPE 1, autrement calculer

$$T_i^* = \frac{t_i}{s_i} \quad , \text{ pour tout } i \in I \text{ avec } s_i > 0.$$

ETAPE 3: Calculer T_i^* pour i avec $s_i=0$ en trouvant une solution réalisable du système linéaire suivant:

$$\sum_{\substack{i\in I \\ s_i=0}} n_{ia}T_i = V_a - \sum_{i\in I^+} n_{ia}T_i^* \quad , \ a\in A, \tag{27}$$

$$T_i \geq 0 \quad , \text{ pour tout } i\in I^0 \text{ avec } s_i=0.$$

et *ARRETER.*

L'algorithme 2 est présenté d'une façon plus technique et est plus proche d'une implantation en forme de programme d'ordinateur que la formulation utilisée pour l'algorithme 1. Il est facile de voir qu'à chaque moment, on a que $s_i = 1 + \sum_{a\in A} n_{ia}\lambda_a$, et que pour le cas où $t_i>0$, $i\in I$, les deux algorithmes sont identiques. Comme on l'a vu plus haut, l'étape 1 correspond encore à une itération de la méthode de descente cyclique au long des axes de coordonnées. Puisque cette fois-ci, la méthode est appliquée pour un problème avec contraintes, on ne peut plus se fier sur les mêmes propriétés de convergence que dans le cas sans contraintes. Théoriquement, il est tout-à-fait possible que l'algorithme 2 s'arrête à un point λ qui n'est pas une solution optimale du problème dual (20). On peut quand même prouver la proposition suivante:

Proposition 6: Si la région réalisable de (27) n'est pas vide lorsque l'algorithme 2 entreprend l'étape 3, la solution trouvée est optimale.

Preuve: Soit λ^* les valeurs de λ au moment où l'étape 3 commence. Evidemment, λ^* est dual-réalisable. Si on réussit à calculer T^*, i.e. si la région réalisable de (27) n'est pas vide, on est assuré que T^* est primal-réalisable. Puisque (T^*,λ^*) satisfont les conditions d'optimalité (22) par

construction, ils sont optimaux. ■

Cela nous indique qu'on peut se servir de (27) pour déterminer si l'algorithme 2 a vraiment convergé à une solution optimale. Toutes nos expériences ont montré que l'algorithme 2 est très robuste et qu'il a toujours convergé vers une solution optimale pour tous les problèmes pratiques qu'on a rencontré jusqu'à date. Pour certaines classes de problèmes, comme par exemple le problème d'estimation d'une matrice O-D à vraisemblance maximale quand les totaux marginaux sont connus (voir plus loin), on peut montrer que (27) a toujours une solution, et dans ce cas, l'algorithme 2 devient globalement convergent.

2.4 Cas spécial des totaux marginaux connus

Un cas spécial important du problème d'estimation d'une matrice O-D de vraisemblance maximale (1) se présente quand, au lieu de connaître les volumes sur les liens, on connaît plutôt les totaux marginaux de la matrice (T_{pq}), $(i \equiv (p,q))$, i.e. les productions et les attractions sont connues. Soit O_p la production totale de l'origine p, $p \in P$, et soit D_q l'attraction totale de la destination q, $q \in Q$. Comme avant, nous faisons l'hypothèse que la matrice (t_{pq}) contient des valeurs entières qui proviennent de distributions de Poisson indépendantes. Le modèle de vraisemblance maximale (8) peut être réécrit comme

$$\text{Min} \sum_{p \in P} \sum_{q \in Q} (T_{pq} - t_{pq} \log T_{pq}) \tag{28}$$

sujet à

$$\sum_{q \in Q} T_{pq} = O_p \quad , \; p \in P, \tag{29}$$

$$\sum_{p\varepsilon P} T_{pq} = D_q \qquad , q\varepsilon Q, \tag{30}$$

$$T_{pq} \geq 0 \qquad , p\varepsilon P,\ q\varepsilon Q. \tag{31}$$

On peut voir facilement que le problème (28) est un cas spécial du problème (8), en notant que la structure de (28) correspond à un réseau "étoile", ayant un sommet central et des liens de chaque origine à ce sommet et des liens du sommet central à chaque destination.

Il est possible de limiter le problème considéré au cas où $O_p>0$, $p\varepsilon P$, et $D_q>0$, $q\varepsilon Q$, sans manque de généralité, puisque s'il existe une origine p avec $O_p=0$, ou une destination q avec $D_q=0$, cela implique que la ligne (ou colonne) correspondante de la matrice (T_{pq}) contient des zéros. On peut donc simplement éliminer les origines (destinations) pour lesquelles $O_p=0$ ($D_q=0$).

Notons que (29) et (30) sont réalisables si et seulement si la somme des productions O_p est égale à la somme des attractions D_q

$$\sum_{p\varepsilon P} O_p = \sum_{q\varepsilon Q} D_q = \sum_{p\varepsilon P} \sum_{q\varepsilon Q} T_{pq}. \tag{32}$$

L'équation (32) nous indique que le premier terme de la fonction objective (28) est constant et peut donc être éliminé pour obtenir

$$\text{Max} \sum_{p\varepsilon P} \sum_{q\varepsilon Q} t_{pq} log T_{pq} \tag{33}$$

sujet à (29) - (31).

Soient α_p, $p\varepsilon P$, les variables duales associées avec les contraintes de production (29) et β_q, $q\varepsilon Q$, celles associées avec les contraintes d'attraction (30). En utilisant un développement similaire à celui suivi

pour obtenir (20), nous pouvons reformuler le dual Lagrangéen de (33) comme

$$\text{Min} \sum_{p\varepsilon P} \sum_{q\varepsilon Q} t_{pq} log \frac{t_{pq}}{\alpha_p+\beta_q} + \sum_{p\varepsilon P} \alpha_p O_p + \sum_{q\varepsilon Q} \beta_q D_q \qquad (34)$$

sujet à

$$\alpha_p + \beta_q \geq 0 \qquad , p\varepsilon P,\ q\varepsilon Q. \qquad (35)$$

Soit T^* une solution optimale du problème primal (33) et soit (α^*,β^*) une solution optimale du dual (34). Les conditions d'optimalité de Kuhn - Tucker s'écrivent alors comme

$$T^*_{pq}(\alpha^*_p + \beta^*_q) = t_{pq} \qquad , p\varepsilon P,\ q\varepsilon Q, \qquad (36)$$

ce qui se simplifie encore plus dans le cas où on a $t_{pq}>0$, soit

$$T^*_{pq} = \frac{t_{pq}}{\alpha^*_p + \beta^*_q}. \qquad (37)$$

Notons que ce modèle, obtenu de l'approche de vraisemblance maximale appliquée à une matrice observée Poissonienne, est différent du modèle d'entropie maximale, aussi connu sous les noms balancement biproportionnel, Fratar (voir Lamond et Stewart 1981) et méthode de Furness (Furness 1970), qui donne lieu à un modèle de la forme

$$T^*_{pq} = e^{\alpha^*_p + \beta^*_q} t_{pq} \qquad , p\varepsilon P,\ q\varepsilon Q. \qquad (38)$$

Voici donc l'énoncé de l'adaptation de l'algorithme 2 au problème d'estimation de matrice O-D avec des totaux marginaux connus (33).

Algorithme 3:

ETAPE 0: Mettre $\alpha_p := 1/2$, $p \in P$, et $\beta_q := 1/2$, $q \in Q$.

ETAPE 1: Pour chaque $p \in P$ faire:

Trouver $\bar{\alpha}_p \geq -\min_{q \in Q} \beta_q$ qui minimise $\left| \sum_{q \in Q} \frac{t_{pq}}{\bar{\alpha}_p + \beta_q} - O_p \right|$.

Mettre $\delta_p := \alpha_p - \bar{\alpha}_p$ et $\alpha_p := \bar{\alpha}_p$.

ETAPE 2: Pour chaque $q \in Q$ faire:

Trouver $\bar{\beta}_q \geq -\min_{p \in P} \alpha_p$ qui minimise $\left| \sum_{p \in P} \frac{t_{pq}}{\alpha_p + \bar{\beta}_q} - D_q \right|$.

Mettre $\delta_q := \beta_q - \bar{\beta}_q$ et $\beta_q := \bar{\beta}_q$.

ETAPE 3: Si $||\delta|| > \varepsilon$ alors retourner à l'ETAPE 1, autrement calculer

$$T^*_{pq} = \frac{t_{pq}}{\alpha_p + \beta_q} \qquad , (p,q) \in I^+.$$

ETAPE 4: Calculer

$$T^*_{pq} = \frac{\bar{O}_p \bar{D}_q}{\bar{T}} \qquad , (p,q) \in I^0, \tag{39}$$

où

$$\bar{O}_p = O_p - \sum_{\substack{q \in Q \\ t_{pq} > 0}} T^*_{pq} \quad , p \in P, \tag{40}$$

$$\bar{D}_q = D_q - \sum_{\substack{p \in P \\ t_{pq} > 0}} T^*_{pq} \quad , q \in Q, \tag{41}$$

$$\bar{t} = \sum_{p\varepsilon\bar{P}} \bar{O}_p = \sum_{q\varepsilon\bar{Q}} \bar{D}_q. \qquad (42)$$

et ARRETER.

Les deux sous-problèmes de minimisation unidimensionnelle dans les étapes 1 et 2 sont équivalents à trouver les zéros des fonctions correspondantes, s'il y en a. Si la fonction ne prend pas la valeur zéro dans l'intervalle considéré, le minimum est toujours atteint à la borne inférieure de l'intervalle, parce que dans ce cas-ci la fonction est monotonement croissante à l'intérieur de l'intervalle.

Proposition 7: L'algorithme 3 converge vers une solution optimale du problème (33).

Preuve: Puisque le problème (33) est un cas spécial du problème d'estimation de matrice (8), et puisque l'algorithme 3 est équivalent à l'algorithme 2 appliqué à (33), nous pouvons utiliser la proposition 6, i.e. il suffit de montrer que l'équation (27) est toujours réalisable. Dans le contexte des totaux marginaux connus, (27) devient

$$\sum_{q\varepsilon\bar{Q}} T_{pq} = \bar{O}_p \qquad , p\varepsilon\bar{P}, \qquad (43)$$

$$\sum_{p\varepsilon\bar{P}} T_{pq} = \bar{D}_q \qquad , q\varepsilon\bar{Q}, \qquad (44)$$

$$T_{pq} \geq 0 \qquad , p\varepsilon\bar{P}, \; q\varepsilon\bar{Q}. \qquad (45)$$

Les productions résiduelles $\bar{O}_p$ et les attractions résiduelles $\bar{D}_q$ sont définies dans (40) et (41). $\bar{P}$ est l'ensemble des origines pour lesquelles on a $\bar{O}_p > 0$ et $\bar{Q}$ est l'ensemble des destinations pour lesquelles on a $\bar{D}_q > 0$. Les contraintes (43) - (45) ont exactement la même structure que les con-

traintes originales (29) - (31), mais elles ne s'appliquent qu'à la sous-matrice $\bar{P}x\bar{Q}$. Le système (43) - (45) est donc réalisable si et seulement si $\bar{O}_p$ et $\bar{D}_q$ satisfont

$$\sum_{p\in\bar{P}} \bar{O}_p = \sum_{q\in\bar{Q}} \bar{D}_q$$

Il est trivial de vérifier que cette condition est toujours satisfaite. Par conséquent, la matrice T^* qui résulte de l'algorithme est primal-réalisable. Puisqu'en plus, (α^*, β^*) sont dual-réalisables et que les conditions d'optimalité (36) sont satisfaites par construction, on peut conclure que T^* est une solution optimale de (33). ■

Notons qu'en général, le système linéaire (43) - (45) a une multitude de solutions réalisables. La solution trouvée dans l'étape 4 de l'algorithme est celle qui maximise l'entropie

$$\text{Max} \sum_{p\in\bar{P}} \sum_{q\in\bar{Q}} T_{pq} log T_{pq}.$$

Bien sûr, toute autre solution réalisable de (43) - (45) mène aussi à une solution optimale de (33).

Après avoir établi la convergence de l'algorithme 3, nous discutons maintenenant, sans aller trop dans les détails, de la résolution des sous-problèmes unidimensionnels qui font partie des étapes 1 et 2 de l'algorithme. Comme nous l'avons déjà mentionné plus haut, le problème principal consiste à trouver la racine de la fonction correspondante. Une méthode efficace pour faire cela est celle de Newton, qui consiste en une suite d'approximations linéaires. Son adaptation mène aux approximations suivantes

dans l'étape 1: $$\bar{\alpha}_p := \bar{\alpha}_p + \frac{\sum_{q\in Q} \frac{t_{pq}}{\bar{\alpha}_p+\beta_q} - O_p}{\sum_{q\in Q} \frac{t_{pq}}{(\bar{\alpha}_p+\beta_q)^2}} \qquad (46)$$

et

dans l'étape 2: $$\bar{\beta}_q := \bar{\beta}_q + \frac{\sum_{p\in P} \frac{t_{pq}}{\alpha_p+\bar{\beta}_q} - D_q}{\sum_{p\in P} \frac{t_{pq}}{(\alpha_p+\bar{\beta}_q)^2}} \qquad (47)$$

La convergence de ces récursions est assurée si on choisit les valeurs initiales

$$\bar{\alpha}_p := \frac{\sum_{q\in Q^{min}} t_{pq}}{O_p} - b^{min} \text{ et } \bar{\beta}_q := \frac{\sum_{p\in P^{min}} t_{pq}}{D_q} - a^{min} \qquad (48)$$

où P^{min} et Q^{min} correspondent aux ensembles d'origines et de destinations pour lesquelles le minimum de α_p et β_q est atteint. Après quelques itérations de l'algorithme 3, les valeurs des variables duales ne changent que très peu, il vaut donc mieux utiliser les valeurs courantes de α_p et β_q comme valeurs initiales de la recherche unidimensionnelle, et de n'utiliser (48) que dans le cas où (46) ou (47) ont généré des valeurs non-réalisables de α_p ou β_q.

t_{pq}	q=1	2	3	4	O_p
p=1	10.000	.000	15.000	5.000	24.000
2	.000	11.000	.000	.000	18.000
3	6.000	9.000	5.000	.000	19.000
4	.000	.000	8.000	4.000	11.000
D_q	19.000	17.000	23.000	13.000	72.000

TABLEAU II-2-1: Un petit exemple

Illustrons maintenant l'algorithme 3 par un petit exemple avec seulement 4 origines et 4 destinations. Les productions O_p, les destinations D_q et la matrice observée (t_i) sont montrées dans le tableau II-2-1.

Le tableau II-2-2 nous montre les valeurs des variables duales après chaque itération, ainsi que la valeur de la fonction objective duale et la valeur du critère d'arrêt $||d||$. Les bornes inférieures actives sont soulignées.

Iter.	a1	a2	a3	a4	b1	b2	b3	b4	OBJ.FCT.	$\|d\|$
0	.500	.500	.500	.500	.500	.500	.500	.500	230.8241	-
1	.750	.111	.353	.391	.177	.905	.354	.022	224.3636	2.042
2	.949	-.022	.548	.778	.091	1.004	.409	.022	222.4997	.855
3	1.035	-.022	.612	.844	.022	.989	.336	.022	222.2466	.373
4	1.090	-.022	.673	.881	.022	.975	.284	.022	222.1411	.219
5	1.110	-.022	.688	.909	.022	.972	.263	.022	222.1265	.087
6	1.119	-.022	.694	.921	.022	.971	.255	.022	222.1240	.036
7	1.122	-.022	.697	.926	.022	.970	.251	.022	222.1236	.015
8	1.124	-.022	.698	.928	.022	.970	.249	.022	222.1235	.006
9	1.124	-.022	.698	.928	.022	.970	.249	.022	222.1235	.003
10	1.125	-.022	.698	.929	.022	.970	.248	.022	222.1235	.001

TABLEAU II-2-2: Variables duales et valeurs de la fonction objective

Enfin dans le tableau II-2-3, on trouve la solution optimale du problème, telle que calculée dans les étapes 3 et 4 de l'algorithme.

T^*_{pq}	q=1	2	3	4	O_p
p=1	8.718	.000	10.924	4.359	24.000
2	1.958*	11.606	.000	4.436*	18.000
3	8.326	5.394	5.281	.000	19.000
4	.000	.000	6.795	4.205	11.000
D_q	19.000	17.000	23.000	13.000	

TABLEAU II-2-3: Solution optimale T^*_{pq} (*: $(pq) \varepsilon \bar{P} \times \bar{Q}$)

Quand on applique la méthode de Furness au même exemple (tableau II-2-1), on s'aperçoit que le modèle biproportionnel (38) qui en résulte n'a pas de solution, puisqu'il y a beaucoup de paires O-D avec $t_{pq}=0$, ce qui rend le problème biproportionnel irréalisable.

2.5 L'inférence statistique

Jusqu'à date, c'est en vain qu'on recherche dans la littérature des méthodes statistiques permettant de tester la cohérence entre le modèle d'estimation de matrice O-D utilisé, les données auxquelles la méthode est appliquée et la qualité de la matrice prédite résultante. Dans la section 2.1, nous avons vu que pour les modèles existants, la matrice $\{t_i\}$ est considérée comme une matrice de référence *constante*, dont on ne connait pas la provenance statistique. Ces méthodes essaient alors de déterminer la matrice $\{T_i\}$ la plus similaire à $\{t_i\}$ parmi celles qui satisfont (2) et (3). L'aspect statistique de ces modèles est donc limité aux mesures de "goodness of fit", comme celles proposées par Smith et Hutchinson (1981), mais il est impossible de faire de l'inférence statistique pour tester la

qualité des estimations résultantes.

Dans les sections précédentes, nous avons toujours pris pour acquis que les éléments de la matrice observée $\{t_i\}$ provenaient de distributions de Poisson avec moyennes $\{T_i\}$ et que la matrice $\{T_i\}$ satisfaisait (2) et (3). On a donc supposé qu'on connaissait a) les *vrais* volumes des liens V_a et b) les *vraies* proportions d'affectation n_{ia}. Dans la plupart des applications, on connaît ce qu'on pense être des estimations raisonnables de ces quantités, mais on est loin d'être sûr qu'on connaît leurs véritables valeurs.

C'est pour cela qu'il est fortement souhaitable d'avoir un test statistique qui nous permette de tester l'hypothèse nulle H0:

H0: $\{t_i\}$ est distribuée selon des lois de Poisson indépendantes avec moyennes $\{T_i\}$ qui satisfont (2) et (3),

contre l'hypothèse alternative

H1: $\{t_i\}$ n'est pas distribuée selon des lois de Poisson indépendantes avec moyennes $\{T_i\}$ satisfaisant (2) et (3).

Une statistique très simple qui nous permet de tester l'hypothèse H0 dans le cas où les proportions d'affectation ne prennent que des valeurs 0 ou 1, est

$$v_a = \sum_{i \in I} n_{ia} t_i . \qquad (49)$$

Dans ce cas, il est facile de vérifier que sous H0, v_a est distribué selon

une distribution de Poisson avec moyenne v_a. On va rejeter l'hypothèse H0 si v_a se trouve hors de l'intervalle de confiance de la distribution de Poisson correspondante.

Il est important de noter que le test basé sur (49) ne s'applique qu'à un seul lien du reseau. Plusieurs tests de ce type, appliqués à différents liens, ne sont pas mutuellement indépendants en général, et l'interprétation des résultats n'est pas une tâche facile.

Dans le cas spécial du problème d'estimation de matrice avec totaux marginaux connus (28), les tests sur les contraintes de productions (ou attractions) sont mutuellement indépendants, grâce à la structure spéciale de (29) et (30). Sous H0, les quantités

$$o_p = \sum_{q \in Q} t_{pq} \qquad , p \in P, \tag{50}$$

sont distribuées indépendamment, selon des distributions de Poisson avec moyenne O_p et

$$d_q = \sum_{p \in P} t_{pq} \qquad , q \in Q, \tag{51}$$

sont distribuées indépendamment avec moyenne D_q. Par contre, les distributions de o_p et d_q ne sont pas indépendantes, parce qu'elles incluent les voyages t_{pq}.

Un autre test, plus global que celui vu plus haut, est basé sur le ratio de vraisemblance ρ, i.e. la valeur maximale de la vraisemblance (6) quand on impose les contraintes (2) et (3), divisées par la valeur maximale de la vraisemblance lorsqu'aucune contrainte n'est imposée. Ce test tient compte des volumes sur tous les liens du réseau et s'applique même pour des

proportions d'affectation n_{ia} non-entières. On sait bien (Wilks, 1962) que la quantité $Y = -2log\rho$ est distribuée asymptotiquement comme χ^2 avec un nombre de degrés de liberté qui correspond au nombre de contraintes non-redondantes dans (2). Puique la solution optimale du problème (8) sans contraintes est $\{T_i\} = \{t_i\}$, on obtient

$$Y = -2log\rho = 2 \sum_{i\in I} (T_i^* - t_i (1+log\frac{T_i^*}{t_i})). \qquad (52)$$

Asymptotiquement, Y est distribué χ^2 avec $|A|$ degrés de liberté, en supposant que (2) ne contienne pas de contraintes redondantes.

Pour le problème avec totaux marginaux connus (28), on obtient de même façon que Y est asymptotiquement χ^2 avec $|P|+|Q|-1$ degrés de liberté, parce que (32) nous indique qu'il y a toujours une contrainte redondante parmi les contraintes (29) et (30). La statistique Y est très similaire, mais n'est pas équivalente, à une mesure de "goodness of fit", telle que proposée par Wilson (1976) et étudiée par Smith et Hutchinson (1981) dans le contexte des modèles de distribution de voyages. Cette mesure, appellée "log likelihood difference" est définie comme

$$LLD = \sum_{p\in P} \sum_{q\in Q} t_{pq} log\frac{t_{pq}}{T_{pq}^*} \qquad (53)$$

Les deux statistiques Y et LLD ne sont pas équivalentes, sauf dans le cas où $\sum_{p\in P} \sum_{q\in Q} t_{pq} = \sum_{p\in P} \sum_{q\in Q} T_{pq}^*$ est satisfait.

La complexité du modèle nous empêche de calculer analytiquement la distribution exacte de la statistique Y. Mais puisque les problèmes qu'on

trouve en pratique sont de taille assez grande et comportent un grand nombre de voyages individuels, on s'attend à ce que la vraie distribution de Y ne soit pas très différente de la distribution asymptotique χ^2. Pour étudier empiriquement la similarité entre la vraie distribution et la distribution χ^2, nous avons utilisé la technique de simulation de Monte Carlo pour générer une série de valeurs y_n qui sont distribuées selon la distribution exacte de Y.

Pour ce faire, nous avons choisi des exemples qui sont constitués de réseaux linéaires (chaînes de liens) comportant 5, 10, 15 et 20 liens, ce qui correspond à 15, 55, 120 et 210 paires O-D. D'abord, nous avons généré une vraie matrice O-D $\{T_i\}$ et les volumes V_a ont été calculés en utilisant (2). Ensuite, une fois pour chaque essai, une matrice $\{t_i\}$ fut générée en utilisant des nombres aléatoires Poissonniens. Le problème (8) fut résolu en utilisant l'algorithme 2, et la valeur y_n fut calculée. La figure II-2-1 montre la comparaison entre les distributions empiriques, obtenues après 1000 essais, et la distribution asymptotique χ^2. La similarité des deux distributions est clairement visible dans la figure II-2-1. En plus, le test de Kolmogorov - Smirnov (Wilks, 1962) a été appliqué à ces données. Dans aucun des exemples, on ne pouvait détecter une différence significative entre la distribution empirique et χ^2 (niveau du test 5%).

La même technique de simulation Monte Carlo a été appliquée ensuite au cas spécial du problème d'estimation de matrice O-D avec des totaux marginaux connus. On a généré 5 vraies matrices $\{T_{pq}\}$, de dimensions $|P|=|Q|=5$, 10, 15, 20 et 25, en utilisant des nombres aléatoires distribués selon une loi exponentielle avec une moyenne de 5 voyages. La figure II-2-2 montre les distributions empiriques, obtenues après 1000 essais,

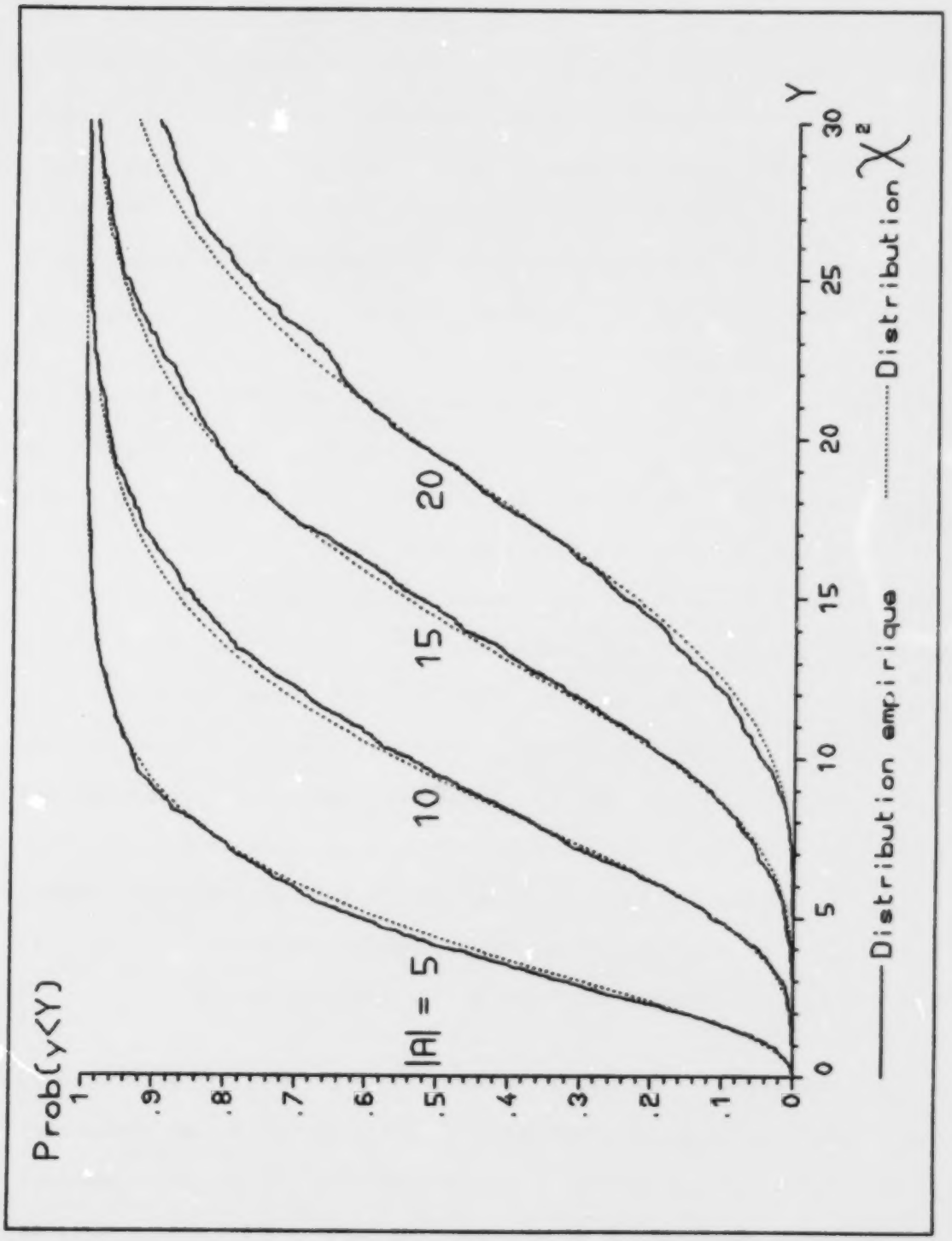
Prob(y<Y)
1
.9
.8
.7
.6
.5
.4
.3
.2
.1
0
|A| = 5
10
15
20
0
5
10
15
20
25
30
Y
Distribution empirique
Distribution χ^2

FIGURE II-2-1

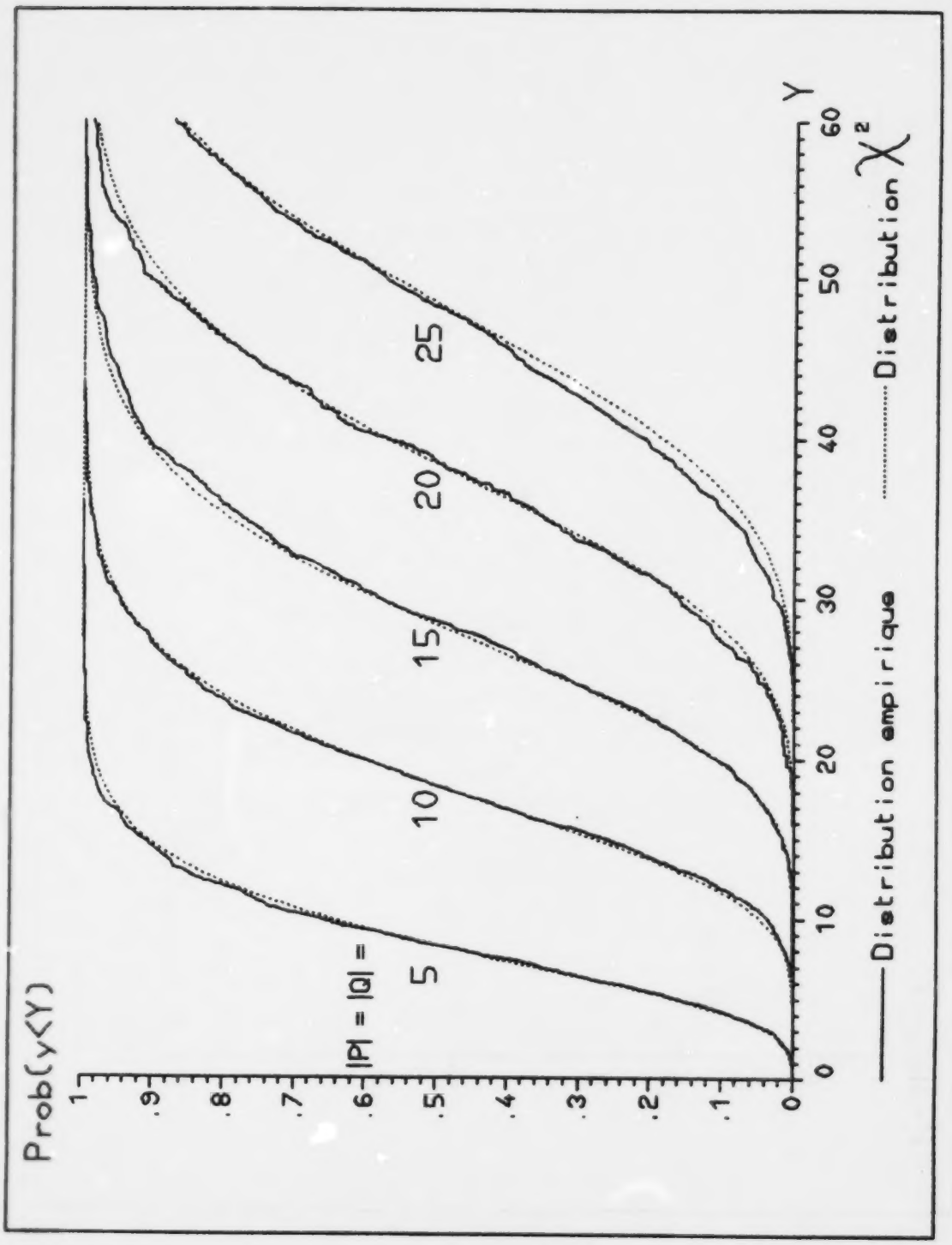
Prob(y<Y)
1
.9
.8
.7
.6
.5
.4
.3
.2
.1
0
|P| = |Q| =
5
10
15
20
25
0
10
20
30
40
50
60
Y
Distribution empirique
Distribution χ^2

FIGURE II-2-2

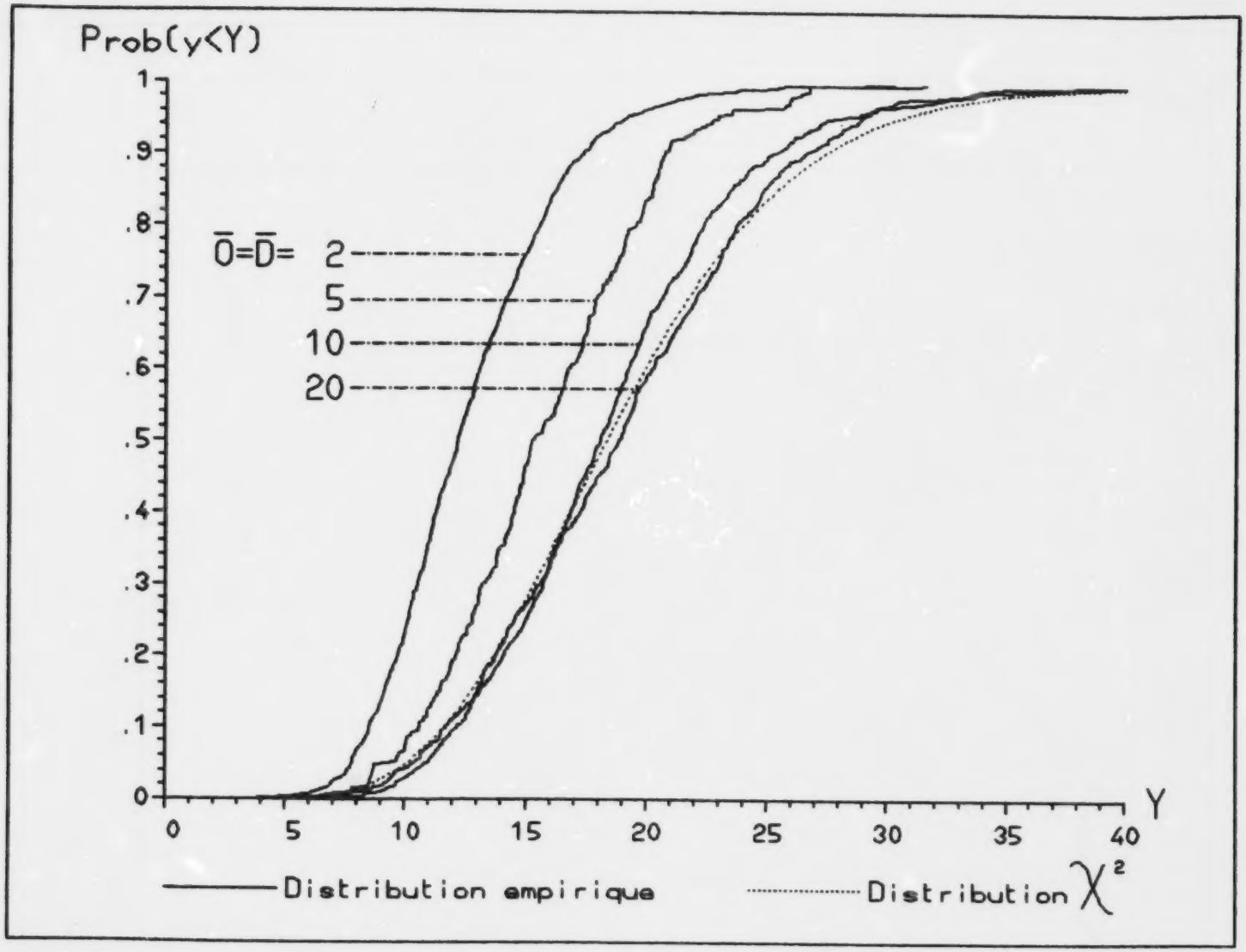
Prob(y<Y)
1
.9
.8
.7
.6
.5
.4
.3
.2
.1
0
$\bar{O}=\bar{D}=$ 2
5
10
20
0
5
10
15
20
25
30
35
40
Y
Distribution empirique
Distribution χ^2

FIGURE II-2-3

comparées aux distributions χ^2 asymptotiques. Comme auparavant, le test de Kolmogorov - Smirnov ne détecte de différence significative dans aucun des 5 exemples.

Toutes nos expériences semblent donc indiquer que la distribution de la statistique Y est très bien approximée par une distribution χ^2, à condition que les volumes sur les liens V_a (ou les productions O_p et attractions D_q) ne soient pas trop petits (disons, plus grands que 10). La figure II-2-3 montre le comportement de la distribution de Y pour des valeurs plus petites que V_a ou O_p et D_q. Dans ce but, on a généré des problèmes (28) avec $|P|=|Q|=10$ en utilisant des productions et attractions moyennes de 2, 5, 10 et 25 voyages. Comme on peut le voir dans la figure II-2-3, l'approximation χ^2 ne tient plus pour les valeurs plus petites que 10. Cela ne pose pas un problème en pratique quand on a à travailler sur des volumes de liens et des productions et attractions beaucoup plus grands que cette valeur critique de 10.

2.6 L'extension du modèle aux volumes observés poissoniens

Dans les sections précédentes, nous avons supposé connus les vrais volumes sur les liens du réseau (ou productions / attractions). Ceci est une hypothèse raisonnable dans le cas où les volumes sur les liens sont comptés sur une base régulière et où on peut estimer leurs moyennes avec une grande précision. Dans d'autres cas, il est possible que les seules données disponibles concernant les volumes (ou productions / attractions) proviennent d'une seule enquête ponctuelle. Dans ce cas, il n'est pas

raisonnable de supposer qu'on connait les vraies valeurs. Il nous faut donc un modèle plus réaliste qui interprète ces volumes observés, dénotés v_a, comme des observations d'une variable aléatoire avec moyenne V_a. Comme pour la matrice observée, on fait l'hypothèse que v_a est distribué selon une loi de Poisson. Si en plus, les observations v_a ont été obtenues d'une façon mutuellement indépendante, et sont indépendantes de l'enquête O-D (t_i), la méthode de vraisemblance maximale nous donne dans le modèle

$$\text{Min} \sum_{i \in I} (T_i - t_i \log T_i) + \sum_{a \in A} (V_a - v_a \log V_a) \qquad (54)$$

sujet à (2) et (3). Evidemment, le modèle (54) est encore un problème de programmation convexe. Les conditions de Kuhn - Tucker qui caractérisent une solution optimale de (54) sont (22) et

$$v_a = V_a^*(1-\lambda_a^*) \quad , a \in A. \qquad (55)$$

Si $v_a > 0$, l'équation (55) peut être utilisée pour substituer λ_a à V_a ou vice versa. Les algorithmes 1 et 2 peuvent alors être modifiés pour résoudre (54). Pour l'algorithme 1, cela est fait simplement en remplaçant V_a par $v_a/(1-\bar{\lambda}_a)$ dans (24) et en changeant l'intervalle admissible pour $\bar{\lambda}_a$ à $(\lambda_a^{min}, 1)$ dans l'étape 1, ce qui nous assure de la non-négativité de V_a.

Cette version modifiée du problème d'estimation de matrice O-D est très utile dans les cas où les volumes observés ne satisfont pas les contraintes de conservation de flôt. L'utilisation du modèle (8) serait impossible dans ces cas, parce que les contraintes (2) ne seraient pas consistantes. La version modifiée du modèle, (54), est toujours réalisable et les volumes prédits, V_a^*, $a \in A$, satisfont toujours les contraintes de conservation de flôt.

Le modèle (54), étant un modèle de vraisemblance maximale, nous permet de calculer la statistique Y comme

$$Y = -2 \sum_{i \in I} (T_i^* - t_i (1 + log \frac{T_i^*}{t_i})) \; -2 \sum_{a \in A} (V_a^* - v_a (1 + log \frac{V_a^*}{v_a})) \qquad (56)$$

qui est asymptotiquement distribuée χ^2 avec $|A|$ degrés de liberté. Nous avons utilisé des simulations de type Monte Carlo pour estimer la distribution exacte de Y pour les mêmes réseaux d'exemples que ceux utilisés dans la section précédente. Les distributions empiriques, ainsi que les distributions asymptotiques χ^2, sont montrées dans la figure II-2-4; et comme auparavant, on n'aperçoit aucune différence significative entre les distributions empiriques et les distributions χ^2.

On peut appliquer la même extension au cas spécial du problème d'estimation de matrice O-D avec des totaux marginaux connus, et on obtient le modèle suivant

$$\text{Min} \sum_{\substack{p \in P \\ q \in Q}} (T_{pq} - t_{pq} log T_{pq}) + \sum_{p \in P} (O_p - o_p log O_p) + \sum_{q \in Q} (D_q - d_q log D_q) \qquad (57)$$

sujet à (29) - (31). Les constantes o_p et d_q correspondent aux productions et attractions observées, obtenues par des comptages indépendants entre eux et indépendants de l'observation de la matrice O-D (t_i). Si t_{pq}, o_p et d_q sont tous positifs, alors on peut exprimer les conditions d'optimalité du problème (57) comme

$$\frac{t_{pq}}{T^*_{pq}} + \frac{o_p}{O^*_p} + \frac{d_q}{D^*_q} = 3 \qquad , p \in P, \; q \in Q. \qquad (58)$$

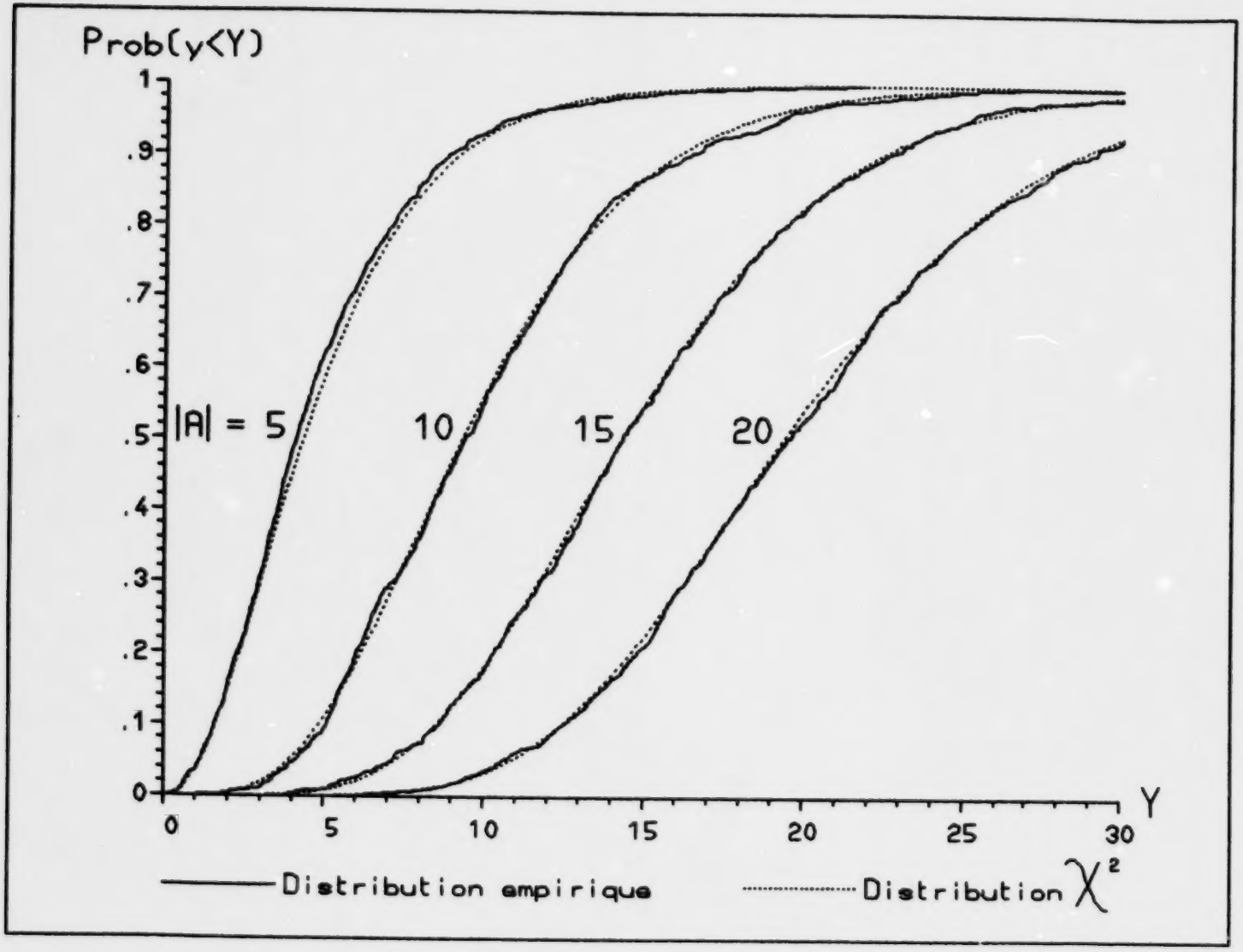
Prob(y<Y)
1
.9
.8
.7
.6
.5
.4
.3
.2
.1
0
|A| = 5
10
15
20
0
5
10
15
20
25
30
Y
Distribution empirique
Distribution χ^2

FIGURE II-2-4

On peut appliquer un algorithme qui est très similaire à l'algorithme 3 pour résoudre cette version du problème d'estimation de matrice avec totaux marginaux connus. L'algorithme suivant suppose que toutes les valeurs observées des productions et attractions sont strictement positives, ce qui est une condition qui est probablement satisfaite dans la grande majorité des cas qui se présentent en pratique. Si on admet aussi des valeurs zéro pour o_p et d_q, on obtient un algorithme légèrement plus compliqué que le suivant.

Algorithme 4:

ETAPE 0: Mettre $\alpha_p := 1/2$, $p \in P$, et $\beta_q := 1/2$, $q \in Q$.

Mettre $\gamma = (\sum_{p \in P} o_p - \sum_{q \in Q} d_q) / (\sum_{p \in P} o_p + \sum_{q \in Q} d_q)$.

ETAPE 1: Pour chaque $p \in P$, faire:

Trouver $\bar{\alpha}_p$ dans l'intervalle $3/2+\gamma \geq \bar{\alpha}_p \geq -\min_{q \in Q} \beta_q$ qui minimise

$$\left| \sum_{q \in Q} \frac{t_{pq}}{\bar{\alpha}_p + \beta_q} - \frac{o_p}{3/2 - \bar{\alpha}_p + \gamma} \right|.$$

Mettre $d_p := \alpha_p - \bar{\alpha}_p$ et $\alpha_p := \bar{\alpha}_p$.

ETAPE 2: Pour chaque $q \in Q$ faire:

Trouver $\bar{\beta}_q$ dans l'intervalle $3/2-\gamma \geq \bar{\beta}_q \geq -\min_{p \in P} \alpha_p$ qui minimise

$$\left| \sum_{p \in P} \frac{t_{pq}}{\alpha_p + \bar{\beta}_q} - \frac{d_q}{3/2 - \bar{\beta}_q - \gamma} \right|.$$

Mettre $d_q := \beta_q - \bar{\beta}_q$ et $\beta_q := \bar{\beta}_q$.

ETAPE 3: Trouver $\bar{\delta}$ dans l'intervalle $-3/2+\max_{p\in P} \alpha_p \leq \bar{\delta} \leq 3/2-\max_{q\in Q} \beta_q$ qui satisfait

$$\sum_{p\in P} \frac{o_p}{1-\alpha_p+\bar{\delta}} - \sum_{q\in Q} \frac{d_q}{1-\beta_q-\bar{\delta}} = 0$$

Mettre $d_0 := \gamma - \bar{\delta}$ et $\gamma := \bar{\delta}$.

ETAPE 4: Si $||d|| > \varepsilon$, alors retourner à l'ETAPE 1, autrement calculer

$$T^*_{pq} = \frac{t_{pq}}{\alpha_p+\beta_q} \qquad , (p,q)\in I^+.$$

$$O^*_p = \frac{o_p}{3/2-\alpha_p+\gamma} \qquad , p\in P,$$

$$D^*_q = \frac{d_q}{3/2-\beta_q-\gamma} \qquad , q\in Q,$$

ETAPE 5: Calculer T^*_{pq}, $(p,q)\in I^o$, en utilisant (39) - (42) et ARRETER.

Comme les autres algorithmes qu'on a développés dans ce chapitre, l'algorithme 4 est aussi basé sur la méthode de descente cyclique le long des axes de coordonnées, ici appliquée au problème dual de (57). On a vu dans la section 3, que cette méthode, appliquée à un problème de programmation convexe avec des contraintes, peut bloquer, i.e. l'algorithme peut se terminer avant qu'on ait trouvé une solution optimale. En effet, si on applique cette méthode directement au problème dual de (57), la convergence de l'algorithme résultant n'est pas garantie. Par contre, si on ajoute explicitement la contrainte redondante de réalisabilité (32), qui donne lieu à la variable duale γ, la convergence de l'algorithme est assurée.

Proposition 8: L'algorithme 4 converge vers une solution optimale du problème (57).

La preuve de la proposition 8 est très similaire à la preuve de la proposition 7 et n'est pas donnée en détail.

Pour le problème (57), la statistique Y est définie comme

$$Y = 2 \sum_{\substack{p\in P \\ q\in Q}} (3T^*_{pq} - t_{pq}(1+log\frac{T^*_{pq}}{t_{pq}})) - 2 \sum_{p\in P} o_p(1+log\frac{O^*_p}{o_p}) - 2 \sum_{q\in Q} d_q(1+log\frac{D^*_q}{d_q}) \qquad (59)$$

et elle est asymptotiquement distribuée χ^2 avec $|P|+|Q|$ degrés de liberté, parce que dans ce cas-ci, (29) et (30) ne contiennent plus une contrainte redondante.

La figure II-2-5 montre les résultats de la méthode de simulation de Monte Carlo pour les mêmes problèmes d'exemple qu'on a utilisés pour obtenir la figure II-2-2. Ni à l'oeil, ni en appliquant le test de Kolmogorov - Smirnov, on ne peut détecter une différence significative entre les distributions empiriques et l'approximation asymptotique χ^2. Notons aussi que la différence d'un degré de liberté par rapport à la figure II-2-2 est clairement visible.

Une des hypothèses qu'on a fait pour dériver (54), était que les volumes observés proviennent des observations qui sont mutuellement indépendantes. La plupart des données disponibles en pratique ne satisfont pas rigoureusement cette hypothèse, parce que les volumes de plusieurs liens ont été comptés dans la même période de temps. Le même voyage peut

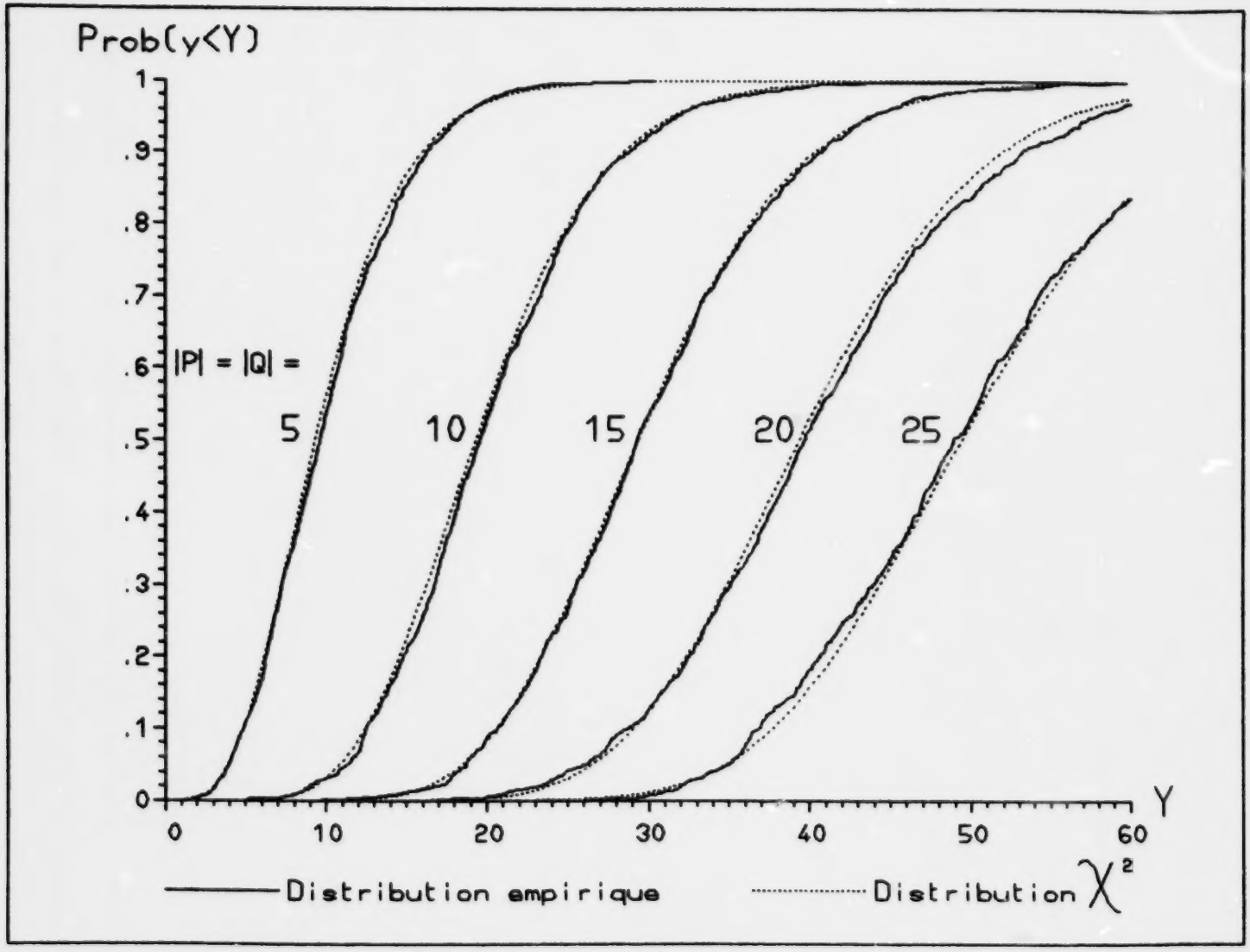
Prob(y<Y)
1
.9
.8
.7
.6
.5
.4
.3
.2
.1
0
|P| = |Q| =
5
10
15
20
25
0
10
20
30
40
50
60
Y
Distribution empirique
Distribution χ²

FIGURE II-2-5

alors être compté sur plus d'un lien, ce qui rend les observation dépendantes. Lorsqu'il est quand même possible d'appliquer le modèle (54) pour des volumes observés quelconques, il faut tenir compte que dans ce cas (54) ne peut plus être interprété comme un modèle de vraisemblance maximale, mais n'en est qu'une approximation.

Le même problème de l'indépendance des volumes observés sur les liens est aussi pertinent dans un autre contexte. Van Zuylen et Willumsen (1981), et plus tard Van Zuylen et Branston (1982), étudient un modèle de vraisemblance maximale pour estimer un ensemble de volumes de liens consistants. Ils font aussi l'hypothèse que les volumes observés suivent des distributions de Poisson et leur modèle a aussi besoin de l'indépendance des observations, même si cela n'est pas indiqué explicitement dans leurs articles.

Pour le cas spécial du problème d'estimation de matrice O-D avec des totaux marginaux connus (57), l'indépendance des observations o_p et d_q est assurée tant que les productions ont été observées indépendamment des attractions. En pratique, ceci est souvent le cas, par exemple, si les productions correspondent à la population active qui habite dans une zone et les attractions correspondent au nombre d'emplois dans une zone, ... ces données étant normalement obtenues par deux enquêtes indépendantes.

2.7 Conclusion

Nous avons proposé un nouveau modèle pour l'estimation d'une matrice O-D, quand les volumes sur les liens du réseau sont connus et une matrice O-D observée est disponible. Nous avons montré qu'en appliquant la méthode de vraisemblance maximale, on obtient un problème de programmation convexe et nous avons développé un algorithme de résolution qui peut résoudre des problèmes de taille réaliste. En parallèle, nous avons considéré séparément le cas spécial du problème d'estimation d'une matrice O-D quand, au lieu des volumes sur les liens, on connaît les productions et les attractions des zones, i.e. les totaux marginaux de la matrice O-D.

Puisque le modèle proposé est basé sur la méthode de vraisemblance maximale, il est possible d'utiliser le test du ratio de vraisemblance pour faire l'inférence statistique sur la cohérence (ou l'incohérence) des données et du modèle appliqué. Ceci distingue ce modèle des autres approches pour l'estimation d'une matrice O-D. Nous avons montré empiriquement, en se servant des simulations de Monte Carlo, que pour des volumes observés (ou productions / attractions) assez grands (>10), la distribution exacte de la statistique Y est très bien approximée par une distribution χ^2. Cette statistique Y peut être utilisée pour tester si les données, soient la matrice observée $\{t_i\}$, les volumes observés V_a et les coefficients d'affectation n_{ia}, satisfont les hypothèses du modèle utilisé. Ce test constitue donc un moyen quantitatif pour mesurer la qualité de la matrice prédite $\{T_i\}$.

Nous tenons à souligner que pour pouvoir appliquer la méthode de vraisemblance maximale, il est nécessaire de donner une interprétation précise à la provenance de la matrice de référence $\{t_i\}$. Dans notre

modèle, nous supposons que les éléments de $\{t_i\}$ sont distribués selon des lois de Poisson. Même s'il est très possible de supposer d'autres provenances de la matrice de référence et d'en déduire d'autres modèles de vraisemblance maximale, le cas de la distribution de Poisson nous semble être le cas le plus important, parce qu'il correspond bien à la plupart des situations où la matrice a été obtenue par des *comptages* de voyages individuels.

Un avantage pratique de notre modèle est dû aux faibles exigences sur les conditions de réalisabilité. Contrairement au modèle d'entropie maximale, pour lequel le fait d'avoir beaucoup d'éléments de $\{t_i\}$ de valeur zéro peut rendre toute la formulation du problème irréalisable, le modèle présenté ici reste toujours réalisable, en autant que les volumes sur les liens V_a soient consistants avec les coefficients d'affectation n_{ia}. Dans la version modifiée du modèle, où les volumes sur les liens sont aussi des observations de variables aléatoires, le problème reste réalisable même quand les volumes observés ne sont pas consistants.

Toutes nos expériences semblent indiquer que pour tout but pratique, la statistique Y peut être traitée comme ayant une distribution χ^2. Jusqu'à date, théoriquement, cela n'est supporté que pour le cas asymptotique, i.e. quand on a observé un grand nombre de matrices $\{t_i\}$ pour chaque expérience. En effet, dans notre modèle, où on n'a qu'une observation de la matrice $\{t_i\}$, il est surprenant que l'approximation asymptotique soit tellement précise. Des recherches plus approfondies seront nécessaires pour établir des résultats sur la distribution exacte de Y et pour étudier plus en détails les conditions nécessaires pour pouvoir appliquer l'approximation χ^2.

II. EMME/2: CONCEPTION ET REALISATION D'UN LOGICIEL INTERACTIF-GRAPHIQUE POUR LA PLANIFICATION DE TRANSPORT URBAIN DE PERSONNES

1. Introduction

Le problème de planification d'un réseau de transport est un problème très complexe qui, à la limite, peut être considéré comme un problème de design optimal. Même si, comme tel, il existe des formulations d'optimisation (voir par exemple Marcotte, 1982), il est impossible, en pratique, de quantifier toutes les contraintes - physiques, budgétaires, politiques et autres - dans un modèle mathématique. De plus, l'objectif d'un exercice de planification n'est pas facile à formuler mathématiquement, parce que, lui aussi, dépend de trop de facteurs non quantifiables. Il n'est donc pas possible - ni souhaitable - de déléguer le processus de planification entièrement à un modèle d'optimisation. Au contraire, il est essentiel que ce processus soit dirigé par le planificateur, puisque son expérience et sa connaissance des détails locaux sont indispensables. Conscient du but du projet et des contraintes imposées, il propose plusieurs *scénarios*, i.e. des variantes de réseaux. Pour pouvoir juger les mérites et les défauts de chaque scénario, il faut pouvoir *prédire* comment les voyageurs vont utiliser le réseau proposé. C'est ici qu'entrent en jeu les modèles analytiques de prédiction ou, comme ils sont appelés plus couramment, les *modèles d'affectation*. A partir d'un réseau

donné et des caractéristiques des voyageurs, ils essaient de prédire les volumes de passagers qui utilisent les divers éléments du réseau, ainsi que le niveau de service qu'ils y trouvent. L'analyse soigneuse de ces résultats permet au planificateur de comparer les conséquences des scénarios. Basé sur ces résultats, il peut alors: soit déclarer un scénario comme solution "optimale" (ou, au moins, acceptable), soit proposer de nouveaux scénarios et recommencer le cycle.

Dans les quinze dernières années, toute une gamme de modèles d'affectation ont été développés: soit pour modéliser le transport routier (comme par exemple TRAFFIC, DAVIS, ICAM, SATURN; voir bas de la page pour les références), soit pour le transport en commun (TRANSCOM, IGTDS, NOPTS, VIPS), soit pour des réseaux multimodaux (UTPS, EMME).

La plupart de ces modèles sont implantés sous forme de programmes de traitement par lots. Bien que leur utilisation se soit avérée très fructueuse, il reste que leur mode d'utilisation est très lourd et l'interprétation des résultats difficile.

Depuis l'introduction des équipement infographiques à prix abordable, le mode de travail interactif-graphique constitue un mode de travail alternatif. L'énorme quantité de données impliquées et leur caractère spatial forment une base idéale pour une approche interactive - graphique. Comme

---

Voici les références aux systèmes mentionnés dans le texte, en ordre alphabétique des acronymes: DAVIS: Tribolt (1979); EMME: Florian (1977), Florian et al. (1979); ICAM: Riepl et Gehner (1978); IGTDS: Rapp (1972), General Motors (1978), Schneider (1978); NOPTS: Rapp et al. (1976), Riepl et Gehner (1978), Douglas (1982); SATURN: Hall et al. (1980); TRAFFIC: Nguyen et Janes (1975); TRANSCOM: Chapleau (1974); UTPS: UMTA/FHWA (1977); VOLVO/VIPS: Andreasson (1976) et (1978), Hasselström (1981).

un des premiers à voir l'importance de cette nouvelle approche, Rapp (1972) a développé le *"Interactive - Graphic Transit Design System"*, ou plus court, IGTDS, un système qui est maintenant supporté par General Motors. De 1974 à 1978, Rapp et son groupe ont développé les systèmes NOPTS et ICAM, deux systèmes très similaires dans leur approche, NOPTS (aussi connu sous le nom TNOP) pour le transport en commun et ICAM pour le transport routier. En même temps, VOLVO développait VIPS, un système interactif-graphique pour la planification du transport public.

Entre temps, plusieurs des systèmes de traitement par lots ont été complémentés par l'ajout de pré- ou post-processeurs, qui permettent au planificateur de préparer les données interactivement et de visualiser les résutats par voie d'affichage graphique (Triboult, 1979; Gotts et Esch, 1982; Grayson, 1982; Dial, 1982).

De 1977 à 1978, le système EMME (acronyme pour *Equilibre Multimodal - Multimodal Equilibrium*) fut développé au Centre de recherche sur les transports. Il s'agissait de la première réalisation d'un modèle d'équilibre pour une affectation multimodale. L'implantation de ce modèle était basée sur les systèmes TRAFFIC, pour le transport routier, et TRANSCOM, pour le transport en commun. Le logiciel résultant, entièrement de type traitement par lots, a servi comme preuve qu'un tel modèle est applicable en pratique. Son utilisation, par contre, s'est avérée trop compliquée pour l'application directe par les agences de planification.

C'est donc en été 1980 que le projet EMME/2 a débuté. Au départ, le but du projet était l'adaptation du logiciel EMME à un mode de travail interactif et graphique. Bientôt, nous nous sommes rendus compte que, si

on voulait réaliser un vrai système interactif-graphique, le simple ajout de pré- et post-processeurs à un programme de traitement par lots existant ne serait pas assez, car le mode de travail interactif-graphique a des implications à tous les niveaux du système.

Dans ce chapitre, nous présenterons le système EMME/2. En particulier, nous nous intéresserons au processus de conception d'un système interactif-graphique. Nous montrerons la multitude de décisions qui doivent être prises pendant le processus de design et nous analyserons les décisions prises dans le cadre de EMME/2.

Dans la section 2, nous étudierons les concepts de base. Tout d'abord, en comparant les deux modes de travail, nous verrons les conséquences profondes qu'a le mode de travail interactif pour la formulation du problème, le fonctionnement du logiciel et la structure de la banque de données. Par la suite, nous développerons un ensemble de concepts de base qui, une fois établi, nous servira comme point de repère pour tous les détails d'implantation. Ces concepts portent sur la structure du logiciel, l'organisation des entrées et sorties, la structure de la banque de données et sur les interactions entre le système et l'usager.

Dans la section 3, nous analyserons les différents types de données qui sont pertinents dans le contexte de planification de réseaux de transport. Nous verrons en détail la définition et la représentation des éléments des trois classes de données: le réseau, les matrices et les fonctions.

Dans la section 4, nous présenterons les aspects graphiques du système

EMME/2. Nous discuterons d'abord des besoins spécifiques associés à un traitement graphique efficace de données de réseau, pour ensuite nous concentrer aux formats d'affichage et aux types d'interaction graphique utilisés dans EMME/2.

Dans la section 5, nous donnerons un aperçu général du système EMME/2 d'un point de vue de l'usager. Ceci est fait en expliquant brièvement le fonctionnement de chaque module.

Dans le section 6, diverses applications de EMME/2, par des agences de planification ou dans le cadre de projets de recherches, seront présentées.

Finalement dans la section 7, nous conclurons par une courte discussion des développements futurs envisagés dans le cadre de EMME/2.

2. Concepts de base

2.1 L'approche interactive-graphique vs. le traitement par lots

Dans l'introduction à ce chapitre, nous avons montré qu'un système interactif - graphique s'avère une base idéale pour le traitement des problèmes de planification de réseaux de transport urbains. Dans cette section, nous allons étudier les conséquences qu'entraîne un mode d'opération interactif. Comme nous le verrons, les implications vont beaucoup plus loin que seulement le fait de permettre une interaction entre l'usager et le système.

Etant donnée l'existence de plusieurs logiciels de traitement par lots qui s'adressent aux problèmes de planification de transport; (voir par exemple: UMTA/FHWA, 1977; Florian et al,1977), une solution attrayante est l'adaptation d'un tel programme pour un traitement interactif ou interactif - graphique. En effet, ce chemin a été emprunté pour plusieurs des logiciels existants (entre autres: Triboult, 1979; Gotts, 1980; Grayson, 1982). L'aspect interactif - graphique de ces systèmes adaptés se limite, en général, à l'ajout d'un préprocesseur pour l'entrée interactive des données et / ou d'un postprocesseur pour l'affichage des résultats. Bien que cela constitue déjà une amélioration considérable, comparé au système de traitement par lots original, un logiciel adapté de cette manière n'exploite pas toutes les possibilités offertes par le mode d'opération interactif - graphique.

Etudions d'abord les caractéristiques principales des deux types de traitement. Dans le tableau 2-1, l'approche interactive - graphique et

l'approche du traitement par lots sont comparées.

Traitement Par Lot:	Système Interactif-Graphique:
Le programme s'adresse à un *problème spécifique*. Ne s'occupe que de la phase de calcul.	Tout un *environnement de travail* est créé. L'usager s'en sert pour définir, résoudre et analyser des problèmes.
Le programme lit un *fichier de données d'entrée*, préparé d'avance par l'usager.	L'usager utilise le système pour l'*entrée* et la *validation* interactive des données.
Le programme génère des *résultats* en forme de fichier de sortie et / ou de rapport imprimé.	Les données d'entrée *et* les résultats sont stockés dans une *banque de données*.
L'usager fait une analyse *manuelle* des résultats.	L'analyse des résultats est faite à l'aide des *outils interactifs - graphiques*.
Le programme essaie de minimiser le *temps d'exécution total*.	L'usager perçoit le *temps de réponse*, qui ne doit pas dépasser une valeur critique.
Les opérations sont arrangées en un ordre *fixe* qui est établi d'avance.	Il n'y a pas d'ordre prédéfini des opérations. La séquence est déterminée par l'usager *pendant* la session.
Tous les aspects du problème sont traités dans un seul passage.	L'usager se concentre sur *un* aspect à la fois.
L'ensemble des données est traité *globalement*.	L'usager choisit *sélectivement* le sous-ensemble d'éléments à considérer.
L'usager doit avoir des connaissances *détaillées* du système d'exploitation.	Une fois que le système est installé, *aucune* connaissance informatique n'est requise.

TABLEAU III-2-1: Traitement par lot vs traitement interactif-graphique.

Comme le démontre le tableau précedent, il y a des différences fondamentales entre ces deux modes de traitement. La raison d'être d'un logiciel de traitement par lots est l'implantation d'un ou d'une suite d'algorithmes afin de calculer les résutats d'un modèle de prévision spécifique. Par contre, dans un système interactif - graphique c'est la création d'un environnement de travail intégré qui est mise au premier plan. Bien sûr, les modèles de prévision constituent toujours une partie vitale du logiciel. Mais ils ne jouent plus le même rôle prépondérant dans l'implantation du système. L'essentiel de l'approche interactive est qu'elle met dans les mains de l'usager un ensemble d'outils. Ces outils correspondent aux tâches auxquelles l'usager se voit confronté:

- l'entrée et la modification de données
- l'affichage graphique de données
- l'impression de données
- le calcul des modèles
- l'analyse locale et globale des résultats
- la comparaison de différents ensembles de données
- la gestion des divers ensembles de données

Du point de vue de l'implantation, ce sont surtout les outils interactifs qui déterminent l'efficacité globale du logiciel, parce que ce qu'on essaie d'optimiser, c'est la communication entre l'usager et le système. En effet, c'est le temps du planificateur qui coûte le plus cher. Le coût du temps d'ordinateur devient de plus en plus négligeable, suivant les développements actuels de prix et performance des appareils informatiques. Ceci ne veut pas dire, qu'il ne faille pas utiliser efficacement les ressources informatiques disponibles. Au contraire, étant donnée la technologie contemporaine, seulement l'utilisation très efficace du matériel permet

d'atteindre le premier but, i.e. un evironnement de travail agréable pour l'usager.

Un aspect important de l'évaluation que fait un usager d'un système interactif, est le *temps de réponse*, i.e. le temps d'attente entre l'*action* de l'usager et la *réaction* du système. Idéalement, ce temps de réponse ne devrait jamais être perçu par l'usager. Dès qu'il le remarque, c'est un signe que le temps de réponse est trop long. Le seuil, en bas duquel une réponse est considérée instantanée, semble être autour d'*une seconde* [Röthlisberger 1980, Scott 1982]. Dans notre contexte, il est impossible de garantir un temps maximal de réponse. Trop de facteurs variables y sont impliqués:

1) la complexité de l'opération demandée
2) la taille du réseau considéré
3) la vitesse du processeur central
4) le temps d'accès aux disques
5) le taux de transmission entre l'ordinateur et le terminal

Voici les délais que nous avons jugés acceptables pour les diverses classes d'opérations:

- La classe d'opérations la plus complexe est évidemment celle des *affectations*. Selon le modèle d'affectation choisi, la taille du réseau et la puissance de l'unité centrale, une affectation peut durer de *quelques minutes* jusqu'à *plusieurs heures*. Par conséquent, dans la majorité des cas, une affectation est préparée interactivement, mais les calculs proprement dits sont effectués en arrière-plan, pendant que l'usager s'occupe d'autres choses.

- Le temps requis pour un *affichage graphique* dépend surtout de la quantité d'information à afficher et du taux de transmission. La plupart des dessins devraient être générés en *une ou deux minutes*. Ce délai semble raisonnable, puisque pendant que le dessin se développe, l'usager peut déjà commencer à se familiariser avec son contenu.

- Une *modification* de la banque de données (ajout, élimination ou modification d'un élément) ne devrait prendre que *quelques secondes*.

- Toutes les *interactions simples*, comme celles qui ont lieu pendant la définition d'une tâche à effectuer, devraient être accomplies *instantanément*, i.e. en moins d'une seconde.

Notons bien que ces temps ne constituent qu'un estimé très grossier de ce que nous considérons acceptable. Ils nous permettent d'analyser la faisabilité des diverses approches considérées.

D'une part, plus le réseau considéré est grand, plus le temps de réponse s'allonge. D'autre part, plus l'ordinateur et l'équipement graphique utilisés sont rapides, plus le temps de réponse décroît. La question de la réalisabilité d'un tel système n'est donc pas *"Quelle taille maximale de réseau un tel système peut-il traiter?"* ou bien *"De quel ordinateur a-t-on besoin pour réaliser un tel système?"*, mais plutôt *"Quelle taille de problème peut-elle être traitée dans tel environnement informatique?"*

L'objectif visé par ce projet, était de démontrer qu'un tel système interactif - graphique est réalisable, tout en utilisant les équipements disponibles, pour le traitement de réseaux de taille moyenne. Comme

exemple d'un réseau de "taille moyenne", nous avons pris le réseau de transport de la ville de Winnipeg. Il s'agit d'un réseau qui comporte environ 150 zones, 1000 noeuds, 3000 liens et 70 lignes de transport en commun. Ce réseau fut choisi, puisqu'un ensemble complet de données et de modèles calibrés était déjà disponible (Florian, 1979). Un intérêt particulier a été mis sur une implantation éventuelle du système résultant sur un *micro-ordinateur*.

Il nous est donc apparu clair que, si le projet devait atteindre cet objectif, il nous fallait plus que le simple ajout d'un pré- et postprocesseur à un logiciel de traitement par lots déjà existant. Il nous fallait plutôt recommencer à zéro et concevoir un nouveau logiciel qui exploite à fond toutes les particularités du traitement interactif - graphique, sans être limité par des contraintes imposées par un système conçu pour le traitement par lots. Evidemment, ceci ne concerne que l'implantation, car le nouveau système sera basé sur le même fondement théorique et permettra l'utilisation des mêmes modèles de prédiction.

En analysant les besoins d'un traitement interactif, on s'aperçoit vite que ce mode de travail a des influences profondes sur toutes les parties essentielles du logiciel, i.e. l'organisation des entrées / sorties, la conception de la banque de données et la structure des programmes. Dans les sections suivantes, nous analyserons ces besoins d'une façon plus détaillée, et nous verrons comment, dans le cadre du système EMME/2, les problèmes d'implantation ont été résolus. Même si on en parle peu dans la littérature, une telle analyse est primordiale pour tout système interactif - graphique, parce que c'est elle qui détermine si le produit final est un vrai système intégré ou simplement un pêle-mêle de programmes.

Comme le lecteur le remarquera bientôt, tous les concepts sur lesquels est basé le système EMME/2, sont d'une simplicité stupéfiante. En général, ceci ne signifie pas que nous n'ayons pas considéré d'autres méthodes plus compliquées, mais que souvent les méthodes les plus simples se sont avérées en même temps les plus efficaces.

2.2 La structure du logiciel

Le logiciel EMME/2 consiste en un ensemble d'environ cinquante *modules*. Dans les pages suivantes, nous étudierons ce qu'est un module de EMME/2 et comment ces modules collaborent pour former un système intégré.

Considérons d'abord le point de vue de l'usager. Pour lui, un module est simplement une partie du logiciel qu'il appelle pour accomplir une tâche particulière. Chaque module a une fonction précise. Il est identifié par un *numéro de module* de la forme *x.yy*. *x* dénote le groupe de modules et *yy* permet de distinguer chaque module à l'intérieur de celui-ci. Pendant une session, l'usager appelle les modules dans un ordre quelconque. Chaque fois qu'un module se termine, l'usager choisit, à l'aide du *menu principal*, le prochain module qu'il désire exécuter. Dans la section 5, on décrira le fonctionnement de chacun des modules.

D'un point de vue informatique, chaque module correspond à un programme indépendant. Il n'y a aucune hiérarchie entre les différents modules. A chaque moment d'une session, il n'y a qu'un et un seul module qui est actif, i.e. qui est en train d'être exécuté. Le menu principal - l'endroit où l'usager choisit le prochain module - constitue lui-même un module, bien que ceci n'apparaisse pas à l'usager. La procédure pour

passer le contrôle d'un module à un autre est définie d'une façon très générale: chaque module peut, à la fin de son exécution, appeler n'importe quel autre module. C'est donc la responsabilité du module d'appeler son module successeur. Tandis que pour la plupart des modules, ce module successeur est automatiquement le module du menu principal, ce dernier établit son successeur par le dialogue avec l'usager.

Chaque module constitue un programme autonome. Aucune information n'est passée directement entre un module et son module successeur. Dès qu'il est appelé, un module lit toutes les informations dont il a besoin de la banque de données. Dans le cas où certaines informations ont été modifiées à l'intérieur du module, celles-ci doivent être réécrites dans la banque de données avant que le prochain module soit appelé. Bien sûr, cette méthode augmente le nombre d'accès nécessaires à la banque de données. Mais c'est grâce à elle qu'on obtient une indépendance complète des modules - un avantage énorme pour le développement et le soutien d'un logiciel de cette taille.

Pour qu'un module puisse passer le contrôle à son successeur, il faut établir un mécanisme de transfert de contrôle. Dans certains systèmes d'exploitation, ceci est possible directement par l'appel à une primitive qui remplace le programme courant par un autre programme. La primitive "exec" du système d'exploitation UNIX en est un exemple. Pour d'autres systèmes d'exploitation, ce transfert ne peut pas être réalisé directement. Dans ces cas, il faut garder un processus additionnel durant toute la session, dont le seul but est d'appeler un module, d'attendre qu'il se termine, d'appeler le module successeur, etc...

Portons maintenant brièvement notre attention à la programmation. Tout le logiciel EMME/2 est écrit en FORTRAN. Ce langage a été choisi pour

- l'efficacité du code généré

- sa capacité de pouvoir traiter de très grands systèmes

- sa disponibilité universelle

Afin de s'assurer que le logiciel résultant soit transférable, sans modifications majeures, à une grande variété d'ordinateurs, nous nous sommes imposés des règles assez strictes sur le style de programmation. En voici les plus importantes:

- On n'utilise qu'un sous-ensemble bien défini du langage FORTRAN, afin que nos programmes soient à la fois compatibles avec les compilateurs FORTRAN IV et FORTRAN '77.

- Seulement le COMMON blanc est utilisé. Il contient dans tous les modules (dans cet ordre):
  - les paramètres globaux
  - les paramètres du scénario courant
  - les vecteurs de données dont le module a besoin

  Les COMMONs nommés ne sont pas utilisés.

- La source de chaque sous-programme contient, dans un format standardisé, la documentation sur son emploi, dans une entête de commentaires.

- Toutes les parties du logiciel qui dépendent de l'installation (ordinateur, système d'exploitation, équipement graphique) sont isolées dans des routines "driver".

Chaque module comprend un programme principal ainsi qu'un ensemble de sous-routines locales, qui sont spécifiques à ce module. Il n'y a aucune dépendance sur l'installation au niveau des modules.

Une bibliothèque de sous-routines générales est partagée par tous les modules. Pendant la phase du chargement d'un module, cette bibliothèque est fouillée pour des références ouvertes. Cette bibliothèque regroupe toutes les routines qui sont utilisées par plus d'un module. Elle comprend environ 150 sous-routines et fonctions dont seulement 19 comportent des dépendances sur l'installation. Seulement ces routines "driver" doivent être adaptées pour transférer EMME/2 d'un environnement à un autre.

Jusqu'à maintenant, le système EMME/2 a été implanté dans les environnements suivants:

- CDC Cyber sous NOS/BE ou NOS/1
- IBM/Amdahl sous TSO
- micro-ordinateur Pixel sous UNIX
- VAX sous VMS (en développement)

2.3 Les entrées / sorties

Dans cette section, nous allons utiliser le terme *entrée/sortie* pour toutes les communications qui ont lieu entre le système interactif-graphique, d'une part, et l'usager, d'autre part. Puisque la banque de données est partie intégrante du système interactif-graphique (i.e. elle ne sert pas directement à la communication système - usager), le transfert de données entre les programmes et la banque n'est pas considéré dans cette section. Une section spécifique sera dédiée à ce sujet.

Le dispositif d'entrée/sortie le plus important dans un système interactif est sans aucun doute le *terminal*. C'est le terminal qui, à l'aide d'un écran et d'un clavier, permet à l'usager de communiquer directement avec le logiciel interactif. Pour qu'on puisse travailler en mode *interactif-graphique*, il faut que le terminal permette, en plus d'une communication alphanumérique, l'affichage graphique, et qu'il soit muni d'un dispositif pour l'entrée graphique. Bien que le terminal graphique constitue l'unité d'entrée/sortie cruciale dans tout système interactif-graphique, il ne faut pas sous-estimer l'importance des autres types d'entrée/sortie. Il est vrai qu'un dessin vaut mille mots. Mais il restera toujours un certain besoin pour l'impression alphanumérique en forme de rapport. Il est vrai que le mode préféré pour l'entrée ou la modification des éléments du réseau est le mode interactif-graphique. Mais un système qui n'offre que ce mode-là devient lent et lourd dans une application où l'information est déjà disponible en forme de fichiers informatisés.

Voici les différents besoins d'entrée/sortie qui ont été identifiés

pour EMME/2:

- communication interactive alphanumérique
- affichage graphique
- entrée graphique
- impression alphanumérique
- lecture de données en format "lot"
- écriture de données en format "lot"
- messages diagnostiques

et voici les différentes unités d'entrée/sortie qui peuvent servir à satisfaire ces besoins:

- terminal alphanumérique et/ou graphique
- imprimante
- traceur de courbes
- tablette graphique (digitaliseur)
- fichiers séquentiels sur disques

Comme on le voit, il y a une multitude de besoins d'entrée /sortie dans un système interactif-graphique, et le terminal ne peut en satisfaire qu'une partie. L'organisation des entrées/sorties doit donc être abordée d'une façon globale, qui tient compte des différentes unités d'entrée/sortie et de tous les besoins de communication.

L'organisation des entrées/sorties qui a été choisie pour EMME/2 est décrite ci-bas. Puisqu'elle est complètement indépendante du contexte des transports, elle pourrait s'appliquer telle quelle à des systèmes interactifs-graphiques dans d'autres domaines.

Au plus bas niveau, on a défini les *unités physiques* où les programmes

peuvent lire ou écrire. Voici celles que connaît EMME/2:

- l'entrée standard (clavier du terminal)
- la sortie standard (écran du terminal)
- la sortie graphique immédiate (écran du terminal graphique)
- l'entrée graphique immédiate
- l'imprimante
- le traceur de courbes
- le fichier séquentiel des données à l'entrée
- le fichier séquentiel des données à la sortie
- le fichier des diagnostiques

L'imprimante et le traceur de courbes peuvent être reliés directement au système ou par l'intermédiaire d'un fichier séquentiel qui accumule l'information jusqu'à la fin de la session de travail, moment où elle sera envoyée à l'équipement correspondant.

Plusieurs des unités physiques mentionnées ci-haut correspondent à des équipements périphériques de l'ordinateur. Pour permettre l'utilisation de différents types de terminaux, de traceurs de courbes et d'imprimantes, une *table des équipements* est maintenue, qui décrit les caractéristiques de chacun des équipements où on peut accéder par une installation particulière. Une description détaillée de la structure de la table des équipements est donnée au chapitre III du manuel de l'usager.

A un plus haut niveau, l'idée principale est de partitionner toute entrée/sortie en *catégories logiques*. Chaque information qui sort ou qui entre du système fait partie d'une (et une seule) catégorie d'entrée/sortie. A chaque catégorie, on associe

- des caractéristiques de format
- des unités physiques qui peuvent être utilisées avec cette catégorie
- un protocole qui décrit les règles que doit suivre tout échange d'information
- un ensemble de sous-routines pour implanter ce protocole

Voici une courte description de chacune des cinq catégories d'entrée/sortie définies pour EMME/2:

DIALOGUE: Comme son nom l'indique, cette catégorie d'entrée/sortie est utilisée pour le dialogue interactif entre l'usager et le système. Le dialogue consiste en une séquence de questions qui sont posées par le système et auxquelles répond l'usager. Pour avoir une interface plus facile avec l'usager et pour rendre le dialogue uniforme et compact, les questions sont divisées en trois types bien définis:

a) La question *YES/NO* est utilisée quand l'usager se voit devant un choix "oui ou non". Son format est caractérisé par un point d'interrogation ("?") à sa fin. Seulement les réponses Y(es), N(o), R(estart) et Q(uit) sont admises. Les deux dernières réponses causent l'interruption du dialogue, soit pour reprendre à partir du début du module courant (Restart), soit pour retourner au menu principal (Quit).

b) La question *SELECT* donne à l'usager le choix parmi plusieurs possibilités. Un menu énumère les options possibles et

l'usager entre le numéro qui correspond à son choix, ou, s'il veut terminer le dialogue, R(estart) ou Q(uit).

c) La question *ENTER*, finalement, a comme but l'entrée d'une ou plusieurs valeurs numériques ou alphanumériques. Toutes les valeurs sont entrées en *format libre*. La réponse, pour ce type de question, peut être donnée sur une seule ligne, ou, pour la spécification d'un ensemble de données, sur une suite de lignes consécutives, terminée par une ligne vide.

En plus des questions, la catégorie "dialogue" est aussi utilisée pour l'affichage de messages qui informent l'usager sur les opérations en cours. Tous les messages qui sont suivis d'un point d'exclamation ("!") sont des messages d'erreur qui avertissent l'usager que le système vient de détecter une inconsistance.

Le format des entrées/sorties de cette catégorie suppose un dispositif d'affichage avec une longueur de ligne d'au moins 80 caractères, et un nombre illimité de lignes. Un terminal à papier continu ou un terminal alphanumérique en mode "scrolling" sont de bons exemples d'un tel dispositif. Chaque séquence de dialogue est précédée de la *chaîne d'initialisation de dialoque*. Cette chaîne est spécifiée, pour chaque type de terminal utilisé, dans la *table des équipements*. La chaîne d'initialisation s'occupe d'effacer l'écran et de mettre le terminal dans le bon mode d'opération.

Le sortie du dialogue est toujours envoyée vers la *sortie standard*, tandis que l'entrée du dialogue provient toujours de l'*entrée standard*.

RAPPORT: Cette catégorie reçoit tout affichage tabulé d'information alphanumérique. Un paquet de telles informations, formatté et paginé, est appelé un *rapport*. Chaque page de rapport porte une entête standard qui indique le module ayant généré cette page, la date et l'usager, ainsi que les descriptions pertinentes de projet, scénario(s) et / ou matrice(s).

Une page de rapport comporte un nombre variable de lignes, dont le nombre maximal est spécifié dans la table des équipements. Chaque ligne d'un rapport est constituée d'au maximum 100 caractères. Cette limite est fixe; la rendre variable en fonction de l'équipement utilisé ne causerait pas seulement une perte d'efficacité considérable, mais poserait aussi des problèmes d'implantation importants. Le maximum de 100 caractères a été choisi pour permettre l'utilisation de formats de papiers réguliers, i.e. 8"1/2 x 11" ou le format métrique A4, qui peuvent être intégrés directement dans un rapport technique.

Chaque page est précédée par la *chaîne d'initialisation de rapport*, qui dépend de l'équipement utilisé et se trouve dans la table des équipements.

Un rapport peut être dirigé soit vers l'imprimante, soit

vers le terminal. Dans le cas de l'*imprimante*, la chaîne d'initialisation de rapport cause l'avancement du papier au début de la prochaine page physique. L'entête de chaque page envoyée à l'imprimante contient un numéro de page. A défaut d'une intervention ponctuelle de l'usager, cette numérotation est continue, sans aucune remise à zéro automatique.

Si un rapport est affiché au terminal, la chaîne d'initialisation de rapport cause l'effaçage de l'écran et, si le terminal le permet, la sélection des petits caractères qui permettent l'affichage de 100 car. par ligne. Une fois la page affichée, la *chaîne d'attention de fin de page* est envoyée à l'écran. Ceci indique à l'usager que la page est maintenant complète et que le système attend un signe pour continuer. L'usager peut demander la page suivante du rapport, s'il en reste, en entrant une ligne vide, ou, s'il veut interrompre l'affichage du rapport, en tapant R(estart) ou Q(uit).

GRAPHIQUE: Cette catégorie est utilisée pour toute entrée et sortie graphique. Les concepts du système graphique et les formats utilisés sont décrits plus bas dans la section 4. Nous nous limitons ici aux aspects de bas niveau.

Les dessins de EMME/2 étant des dessins de vecteurs, le minimum requis d'un équipement graphique est qu'il sache dessiner un vecteur reliant deux points sur l'écran. EMME/2 n'exige pas de résolution minimale *per se*. Pour avoir une

image lisible, une résolution de 512 par 512 pixels semble constituer une borne inférieure. Il n'est pas nécessaire que l'équipement puisse générer des caractères alphanumériques, puisque le système graphique est muni d'un générateur de caractères par logiciel. Si l'équipement graphique permet des opérations plus complexes que le dessin d'un simple vecteur (par ex. génération de caractères, patrons de lignes, couleurs, etc.), ceci est indiqué dans la table des équipements, afin que le système graphique puisse en prendre avantage.

Avant l'affichage d'un graphique, la *chaîne d'initialisation graphique* est envoyée pour effacer la page. Une *chaîne d'attention pour l'entrée graphique* précède toutes les opérations qui utilisent le dispositif d'entrée graphique.

Un dessin peut être acheminé vers le terminal graphique ou vers le traceur de courbes.

TABLE: La catégorie d'entrée/sortie "table" permet de lire (ou écrire) des données en mode de traitement par lot (anglais: batch). Ce mode non-interactif est surtout utilisé pour l'entrée des données préparées à l'extérieur de EMME/2. Il constitue aussi le moyen de tranfert de données entre deux banques de données différentes de EMME/2. Finalement, on se sert de tables afin de pouvoir utiliser des données contenues dans une banque de données EMME/2 avec des programmes externes.

Une table est un fichier séquentiel contenant des lignes

en format de carte perforée: un maximum de 72 caractères de données, suivis optionnellement par 8 caractères de commentaire.

La première ligne d'une table est appelée la *carte T*; elle contient la lettre "T" suivie de l'identificateur d'un type d'élément et, optionnelement, du mot-clef "INIT". La carte T indique le type des éléments qui sont contenus dans la table, et s'il s'agit d'une image complète qui remplace tous les éléments existants ou, en absence de "INIT", qu'il s'agit d'une modification de l'ensemble des éléments existants.

Les *cartes éléments* constituent le corps de la table. Chaque ligne indique l'addition, l'élimination ou la modification d'un élément. Une des lettres "A", "D" ou "M" est suivie de l'identificateur de l'élément et de ses attributs pertinents. Un espace dans la première colonne indique qu'il s'agit d'une continuation de la ligne précédente. Sauf pour la lettre de l'opération, qui se trouve en première colonne, toutes les données suivent les conventions du *format libre*.

La *fin de la table* est indiquée par une ligne vide, ou, en l'absence de celle-ci, par une fin de fichier physique.

Une table peut être preparée par l'usager, ou elle peut être générée par le système EMME/2. Dans ce dernier cas, on parle de "faire un punch", en poursuivant l'analogie avec des cartes perforées. Une table produite de cette façon contient

une description complète des éléments sélectionnés, en forme de cartes d'additions. Ceci permet d'utiliser la table plus tard pour ajouter ces éléments à une différente banque de données EMME/2.

Les tables générées par le système sont dirigées vers le *fichier de tables à la sortie*. Les tables qui sont lues par le système se trouvent dans le *fichier de tables à l'entrée*. Lorsqu'une table est lue, un rapport (voir plus haut) est généré qui contient les cartes lues, ainsi que les messages d'erreurs, s'il y a lieu.

Une description complète des tables séquentielles d'entrée et de sortie se trouve dans le chapitre V du manuel de l'usager.

DIAGNOSTIC: Cette dernière catégorie n'est définie que pour la sortie. Son seul but est de donner des diagnostics précis à chaque fois qu'une erreur est détectée. Normalement, quand l'usager a commis une erreur, il reçoit un message d'erreur sous la forme d'une phrase qui lui explique l'erreur. Un point d'exclamation ("!") sert à distinguer ces messages d'erreurs des autres messages. Dans la plupart des cas, ce message d'erreur est suffisant pour permettre à l'usager de corriger l'erreur et de continuer son travail. Afin de garder une trace de toutes les erreurs et de pouvoir reconstruire les détails, si cela devient nécessaire, un diagnostic est envoyé au *fichier des diagnos-*

tics chaque fois qu'un message d'erreur est envoyé à une autre catégorie d'entrée/sortie (dialogue, rapport ou graphique). Il contient le numéro de module, sa date de compilation, le mode d'opération (interactif ou lot), le scénario courant, un numéro d'erreur et trois paramètres qui décrivent les circonstances précises de l'erreur. Le numéro du module et le numéro de l'erreur permettent à l'usager de trouver une description de l'erreur ainsi que la signification des paramètres dans l'appendice B du manuel de l'usager.

Le fichier des diagnostics s'est avéré très utile. C'est souvent un recours à ce fichier qui permet d'expliquer les erreurs "inexplicables".

2.4 La banque des données

Comme on l'a vu dans l'introduction, un des facteurs principaux qui rendent difficile le traitement des problèmes de planification de réseaux de transport, se trouve dans l'*énorme quantité de données* nécessaires pour définir le problème. Le problème ne réside pas que dans la quantité de données, mais aussi dans leur structure relativement complexe.

La description d'un *réseau de transport*, par exemple, comprend la description de centroïdes, noeuds, liens, modes, véhicules, lignes de transport en commun, segments de lignes, intersections et virages. La description de chaque élément comporte son identificateur, ses attributs caractéristiques, ainsi que les relations qui existent entre cet élément et d'autres. Tandis que les données de réseau constituent la partie "offre de

transport", ce sont les *données matricielles* qui représentent la partie "demande". Ces dernières sont, comparées aux données de réseau, relativement peu complexes dans leur structure. Mais par ailleurs, elles sont plus lourdes en termes de quantité.

Un autre facteur, aussi important que le premier, qui contribue à la complexité des problèmes considérés, est la *quantité énorme de résultats* qui sont générés par les modèles.

Comment organiser efficacement cet ensemble géant de données? Où mettre les résultats, une fois qu'ils sont calculés?

L'approche traditionnelle à ces questions est d'assembler toutes les données d'entrées en un ou plusieurs grands fichiers séquentiels, qui constituent l'entrée au programme. Les modèles de prévision y sont appliqués et les résultats sont envoyés vers un autre grand fichier séquentiel, le fichier des résultats. En fait, c'est ce type d'approche qui est utilisé dans la majorité des logiciels pour la planification de réseaux de transport. Puisque tous ces programmes sont conçus pour le traitement en lot, cette méthode de traitement de données fonctionne très bien. Si, par contre, on essaie d'utiliser cette approche dans le contexte interactif-graphique, on constate très vite qu'elle est tout à fait impraticable. Dans un programme en lot, on sait d'avance exactement ce qui est à faire. Il est donc possible de disposer les données d'entrée dans l'ordre où elles seront utilisées. Un système interactif ne peut pas anticiper l'ordre dans lequel vont se dérouler les différentes tâches. C'est l'usager qui a un contrôle presque complet sur la suite des événements. Etant assis devant le terminal, il s'attend à une réponse instantanée; il est hors de question

d'aller lire toutes les données requises pour l'opération demandée dans des fichiers séquentiels.

On voit donc le besoin d'organiser les données d'une façon plus efficace, qui permette un accès rapide et sélectif aux éléments requis pour accomplir une tâche. Cette organisation interne des données est faite sous la forme d'une *banque de données*, i.e. un fichier-disque à accès direct qui contient les éléments, leurs attributs et leurs relations dans une forme binaire, qui permet un échange d'information efficace et sélectif.

Depuis peu, il est question de *banques de données relationnelles*. Plusieurs logiciels de gestion de banques de données relationnelles sont disponibles sur le marché. La question se pose alors: pourquoi ne pas utiliser un tel système pour la gestion des données de EMME/2? Certes, ces systèmes sont tous capables de gérer le genre de données qu'on retrouve dans EMME/2. Est-ce que leur utilisation sera efficace? Pour répondre à cette question, il faut se rappeler le but pour lequel ces systèmes ont été conçus. En premier lieu, ils sont destinés à des situations où les types de données et leur hiérarchie sont *dynamiques*; i.e. de nouveaux types d'éléments peuvent être ajoutés dynamiquement et les relations hiérarchiques entre ces types peuvent changer pendant la vie d'une banque de données. En second lieu, un système de banque de données relationnelle fournit une méthode très efficace pour la recherche d'un élément *aléatoire* pour accéder à *tous* ses attributs et éléments reliés.

Comment cela se compare-t-il avec les besoins d'un logiciel de planification de réseau comme EMME/2? Pour ce qui est du premier point, il est à noter que les types d'éléments traités par le programme sont connus

d'avance et leur hiérarchie est inhérente au problème même. On est donc en présence de données qui ont des structures et relations *statiques*; ce ne sont que leurs dimensions et leurs valeurs qui sont sujettes à changements. Quant au deuxième point, on s'aperçoit vite, qu'on désire l'accès à un élément particulier dans très peu de situations seulement. La grande majorité des accès est faite *systématiquement* pour tout un *ensemble d'éléments*. En plus, pour chaque accès que l'on fait, on connait exactement les attributs qui seront nécessaires; il s'agit en général de seulement quelques-uns.

Prenons par exemple le dessin du réseau de base. Il faut balayer systématiquement les noeuds. Pour chacun, il nous faut connaître ses coordonnées X-Y ainsi que la liste des liens sortants et leurs noeuds terminaux. Le fait de traiter l'accès à chaque noeud et chaque lien comme un accès aléatoire indépendant pourrait facilement être une tache qui demande des milliers d'accès physiques à l'unité de disques, causant des délais intolérables pour l'usager. Le même raisonnement s'applique aussi bien à l'autre type important de données, les données matricielles. Là aussi, il s'agit presque toujours d'accès systématiques à toute ou à une partie d'une matrice.

Il y a un domaine, par contre, où l'utilisation d'un système de banque de données relationnelle aurait des avantages. Il s'agit de la modification ponctuelle des données: les éditeurs interactifs de réseau et de matrices. Dans ce cas, l'usager demande l'accès à un élément en particulier, qu'il désire modifier selon ses besoins. Cependant, ce genre d'opérations constitue une partie assez faible des accès aux données. De plus, puisqu'il s'agit de modifier un élément à la fois, même une méthode

inefficace est tolérable. Le temps de réponse ne dépassera jamais quelques secondes.

Le recours à un système de banque de données commercial a donc été rejeté dans le contexte de EMME/2, et il nous a fallu trouver une autre approche, qui réponde adéquatement à nos besoins particuliers. Dans les prochaines pages, nous abordons les concepts qui ont été utilisés dans la banque de données de EMME/2. Cet aperçu ne touchera que les idées générales; nous n'aborderons pas directement les structures particulières qui sont employées pour la représentation de chaque type d'élement - ceci sera traité en détail dans la section 3 de ce chapitre.

Du point de vue du système d'exploitation, la banque de données de EMME/2 est simplement un grand fichier à accès direct. Selon l'implantation, ce fichier est adressable par enregistrement (par exemple pour la version IBM / Amdahl), par mot (CDC Cyber) ou par octet (systèmes UNIX).

Du point de vue des modules de EMME/2, la banque de données constitue un ensemble de fichiers logiques, appelés les *fichiers internes*. Chaque fichier interne comporte un nombre donné d'enregistrements à taille fixe. Il est identifié par un numéro unique, et sa dimension est définie par le nombre d'enregistrements, leur taille et le type d'information qu'il contient, i.e. nombres entiers, nombres réels ou caractères alphanumériques.

L'interface entre la représentation interne des données et le format externe de la banque est accomplie par un ensemble de routines d'accès-disques, qui font partie des sous-routines de base de EMME/2.

Une importante caractéristique, particulière à la banque de données de EMME/2, est que les données y sont disposées d'une façon *verticale*. Au lieu d'accumuler dans un enregistrement les valeurs de tous les attributs associés avec un élément (organisation horizontale), un enregistrement de la banque de données de EMME/2 contient les valeurs d'un attribut particulier pour tous les éléments pertinents (organisation verticale). La figure III-2-1 illustre ce concept.

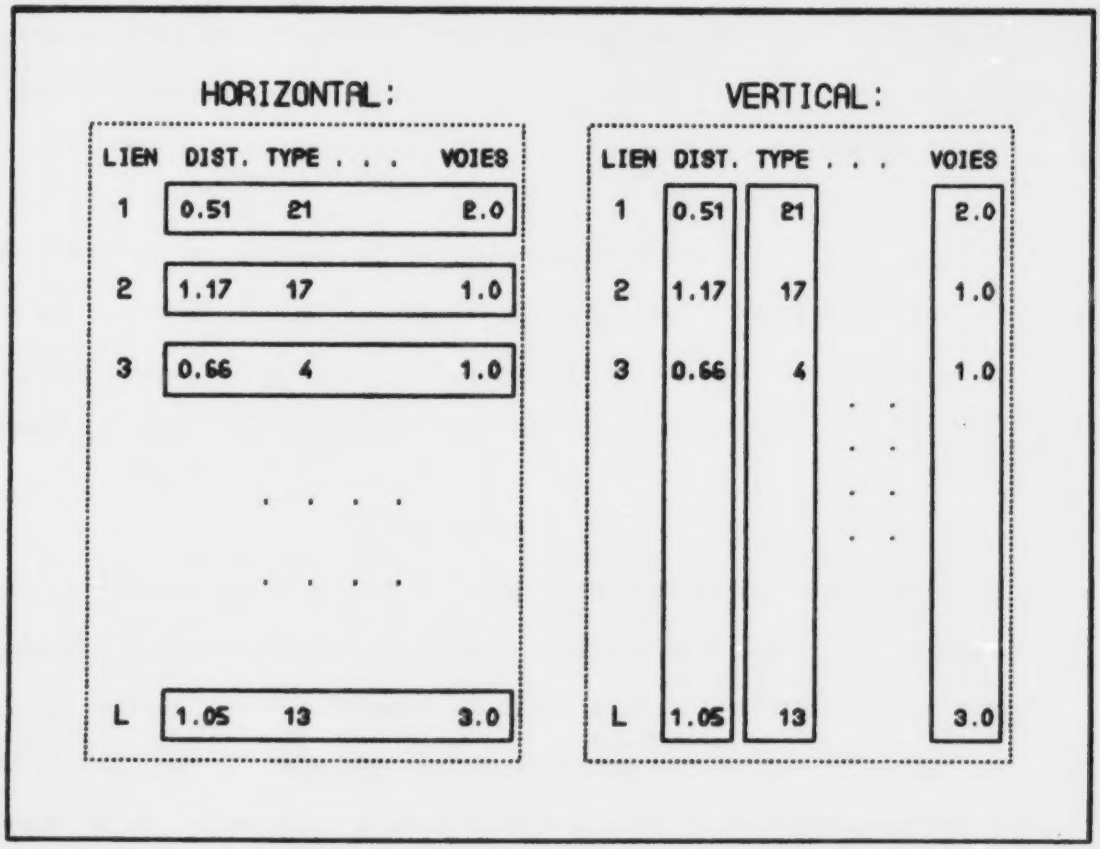

FIGURE III-2-1: Organisation verticale vs. horizontale.

L'emploi d'une organisation verticale comporte plusieurs avantages dans notre contexte:

- Un enregistrement ne comporte que des données homogènes, i.e. du même type. Il n'y donc pas de nécessité pour des structures composées.

- On réduit radicalement le nombre d'appels à la routine d'accès-disques, et par conséquent, le nombre d'accès physiques à l'unité de disques. Ceci découle du fait qu'on a des milliers d'éléments dans la banque, mais peu d'attributs par élément.

- L'organisation verticale nous permet de ne lire que les attributs dont on a vraiment besoin.

Comme on l'a déjà mentionné, les relations entre les divers types d'élément sont fixes, puisqu'elles sont inhérentes au problème. EMME/2 en prend avantage en ne stockant pas cette information: elle est contenue implicitement dans la signification des éléments.

Une fois qu'on a établi le format de la banque de données, il faut décider quelles sont les données qui vont y être contenues. Dans l'approche interactive-graphique, le traitement n'est pas fini une fois que les résultats d'un modèle de prévision ont été calculés. Au contraire, c'est maintenant que commence un aspect important d'un traitement interactif-graphique, l'analyse des résultats. Les données qui sont le résultat d'un modèle deviennent donc les données à l'entrée de la phase subséquente, soit l'analyse interactive-graphique, soit une autre étape d'un modèle de prévision. Il est donc clair que les résultats doivent être stockés comme une partie intégrante de la banque de données. Un autre avantage de procéder de cette manière, est qu'on sépare complètement la phase de calcul de la phase d'analyse. Un modèle de prévision complexe,

qui demande beaucoup de temps d'exécution, peut alors être exécuté comme une tâche non-interactive, en "background", tout en gardant la possibilité d'une analyse interactive-graphique plus tard.

Un autre principe concernant le contenu de la banque de données est appliqué strictement pour EMME/2. Il s'agit de la *non-redondance* des données. Comme cela deviendra évident dans la section 3, où nous aborderons la représentation des divers types d'éléments, on évite autant que possible de stocker des données redondantes dans la banque. En faisant cela, on augmente considérablement la robustesse de la banque de données (*). Au lieu de stocker des données redondantes, on les recalcule à chaque fois qu'on en a besoin dans l'exécution d'un module. Bien sûr, ceci augmente légèrement les temps de calcul. Mais, à notre avis, c'est le seul moyen acceptable pour éliminer la possibilité d'avoir des données inconsistantes dans la banque.

Prenons un exemple: la longueur du trajet d'une ligne de transport en commun est une donnée redondante; elle peut être obtenue comme la somme des longueurs des liens parcourus. On pourrait calculer la longueur de chaque ligne de transport en commun, lorsqu'elle est définie, pour la stocker comme un attribut de la ligne. Ceci nous éviterait d'avoir à refaire ce calcul lorsqu'on a besoin de la longueur du trajet d'une ligne. Vrai? Supposons maintenant que l'usager, après avoir entré les lignes de transport en commun, désire changer la longueur d'un lien. Déjà, on rencontre un problème d'inconsistance entre les données. Voici quatre possibilités

(*) A première vue, ceci peut sembler paradoxal, car, dans d'autres contextes, on fait exprès pour introduire la redondance dans les données pour augmenter la robustesse.

de se tirer de ce problème:

a) On permet à l'usager de faire cette modification et on ne s'occupe pas de l'inconsistance que cela pourrait créer - pour finalement se retrouver devant un désordre complet.

b) On interdit à l'usager de faire cette modification. Résultat: l'usager est forcé de suivre un ordre très strict dans la définition des éléments. Une fois que les éléments de plus haut niveau (ligne de transport en commun) sont créés, il lui est interdit de toucher les éléments de plus bas niveau (liens). Cette aproche remet en question beaucoup des avantages d'un traitement interactif-graphique. L'usager n'est plus le maître - il est devenu esclave.

c) On permet toute modification. Après chaque modification, on recalcule systématiquement toutes les données redondantes qui pourraient être affectées, et on les réécrit dans la banque. Une toute petite première modification peut donc entraîner beaucoup de modifications secondaires, qui elles, causent des modifications tertiaires, ...

d) On ne stocke pas de données redondantes. La méthode de notre choix. Le désavantage qu'on doive dépenser un peu plus d'effort de calcul pour regénérer ces données chaque fois qu'on en a besoin, semble mineur comparé aux conséquences désastreuses des autres "solutions".

Un autre problème se pose dans la gestion d'une banque de données: l'intégrité des données qui y sont contenues. Une fois que la banque est remplie, comment s'assurer de la validité de toutes les données et de la consistance des relations entre les données? Et, au cas où on détecterait

une inconsistance, que faire pour la corriger?

Posées de cette façon, ces questions sont extrêmement difficiles, sinon impossibles, à résoudre. Pour savoir si le contenu d'une banque de données est consistant, il faut faire des tests très complexes, qui comparent tout élément avec tout autre, et prévoir tous les types possibles d'inconsistance - une tâche très difficile à programmer et très gourmande en temps d'exécution. Même si on arrive à franchir cette étape, que fait-on lorsqu'on s'aperçoit d'une inconsistance? Sans connaître l'histoire du développement du contenu de la banque, il est impossible de déterminer a posteriori la cause de l'erreur. Par conséquent, on ne peut prendre les bonnes mesures de correction.

La seule façon simple et élégante de résoudre ce problème est de l'*éviter*. On va simplement *exiger* que la banque soit *toujours* dans un état *consistant*.

Toute modification de la banque de données sera entreprise seulement *après* qu'on ait testé sa validité et qu'on se soit assuré qu'elle ne causera aucune inconsistance. C'est la responsabilité du programme qui veut faire la modification de faire d'abord tous les tests nécessaires. La modification du contenu de la banque ne sera commencée que lorsqu'on n'aura détecté aucun problème.

Plusieurs conséquences importantes découlent de ce concept:

1) Une modification est soumise à des tests complets immédiatement après que l'usager l'ait demandée. Soit que le système détecte une erreur maintenant - et l'usager peut la corriger tout de suite, soit qu'il

n'en détecte pas - et la modification complète est effectuée instantanément. Il est donc impossible que cette modification donne lieu à des messages d'erreurs plus tard, quand l'usager est déjà passé à autre chose.

2) Les vérifications qui doivent être faites sont très simples. Il suffit de tester, pour chaque modification, si les valeurs des attributs sont valides et si les relations entre l'élément modifié et les autres sont consistantes. Si une erreur est détectée, par définition, c'est l'opération actuelle qui est fautive. On n'a donc jamais à chercher "le coupable".

3) Un programme peut toujours se fier au contenu de la banque. Aucun test ne doit être fait après avoir lu des données de la banque. Dans notre contexte,où les lectures sont beaucoup plus fréquentes que les modifications, ceci constitue un facteur important pour maintenir l'efficacité des programmes.

4) Le jeu d'opérations pour modifier la banque est limité aux opérations simples: addition, élimination et modification d'un élément.

5) Une seule modification peut être effectuée à la fois. Chaque modification doit être complétée sans délai. Une fois terminée, la banque est retournée dans un (autre) état consistant.

6) La vie d'une banque de données correspond à une suite de transitions d'un état permis à un autre état permis. Par conséquent, une banque de données vide, l'état initial, doit correspondre à un état permis.

7) Les points 5 et 6 impliquent une *hiérarchie* locale entre les éléments. Un élément peut seulement être ajouté à la banque de données si tous les éléments auxquels il fait référence sont déjà définis. Un élément peut seulement être éliminé de la banque de données si aucun autre élément n'y fait référence.

Nous sommes conscients que ce concept est plutôt conservateur. On peut trouver des cas spéciaux pour lesquels il serait possible de contourner certaines contraintes imposées par les points 1 à 7 ci-haut, sans courir le danger d'une catastrophe. Nous avons résisté jusqu'à présent à ces tentations; nous jugions qu'on ne devrait pas abandonner des principes simples et clairs qui s'appliquent universellement pour de petits gains d'efficacité ici et là.

Enfin, il nous reste à décider l'*ordre* dans lequel les éléments vont être disposés dans la banque de données. Comme on l'a déjà expliqué, l'opération la plus fréquente dans EMME/2 est celle d'un parcours systématique d'un certain ensemble d'éléments. Toute utilisation de structures dynamiques (listes, arborescences, sauts, ...) qui permettent une mise à jour très efficace, cause un ralentissement considérable dans le cas d'un parcours systématique, à cause du nombre élevé d'accès-disques. La structure la plus efficace pour un parcours systématique est en même temps la structure la plus simple possible: la table séquentielle. Son efficacité, par contre, est conditionnelle au fait que l'ordre des éléments dans la table coïncide avec l'ordre de parcours désiré. Pour cette raison, les fichiers internes de la banque de données de EMME/2 sont composés, en plus grande partie, de *tables séquentielles ordonnées*, triées selon l'identificateur des éléments correspondants. Cette structure demande plus

d'effort pour la mise-à-jour, mais ceci est plus que compensé par les avantages qui en découlent pour la recherche d'éléments et pour le parcours systématique:

- La recherche d'un élément particulier peut être effectuée par la méthode de bi-section, en O(log N) opérations.

- Le besoin de trier les sorties est complètement éliminé. Un simple parcours dans l'ordre dans lequel les éléments sont stockés résulte automatiquement en des sorties triées, qui peuvent être acheminées directement vers l'usager.

- La banque de données ne contient aucune hystérèse. Le contenu physique de la banque de données ne dépend point de l'ordre des manipulations, mais seulement des éléments spécifiques et de leurs attributs. Deux banques qui contiennent les mêmes éléments sont physiquement identiques, peu importe l'histoire de leur développement.

- Si l'espace mémoire est limité et qu'on ne puisse se permettre de lire tous les éléments en une seule opération, une lecture par tampon est possible, sans perte significative de vitesse.

- La comparaison de structures complexes est facile (par exemple, la comparaison de la topologie de deux réseaux). Il s'agit simplement de les traverser en parallèle. Avec d'autres méthodes de stockage, ce problème peut devenir très difficile.

En résumé, l'organisation de la banque de données est basée sur un petit ensemble de concepts fondamentaux:

- fichiers internes organisés verticalement

- données de base et résultats stockés dans la banque

- toutes les données redondantes ne sont pas stockées dans la banque, mais sont recalculées à chaque fois qu'on en a besoin

- la détection d'erreur est faite à l'entrée

- les données sont stockées en ordre, triées selon les identificateurs des éléments

Tous ces concepts sont très simples et probablement aucun n'est vraiment nouveau. La plupart ne reflètent que le bon sens. Néanmoins, ils ont joué un rôle crucial dans le développement de EMME/2. Ils furent énoncés explicitement dans la toute première phase du développement. Leur application consciente et stricte simplifia considérablement la conception du reste du système et aida à le rendre cohérent.

2.5 L'interaction entre l'usager et le système

Le terme *système interactif* s'applique à tout logiciel qui permet une communication bidirectionnelle entre l'usager, d'une part, et le logiciel, d'autre part, *pendant* l'exécution du programme. De plus, dans le cas d'un *système interactif-graphique*, au moins une partie de cette communication est accomplie par voie de représentations graphiques. Cependant, ces termes n'impliquent pas un déroulement particulier de la communication interactive. Une tâche importante dans la conception d'un système interactif-graphique est donc d'établir les règles de base qui gouvernent

la communication entre l'usager et le système. Dans cette section, nous énoncerons les concepts de base utilisés dans le système EMME/2.

Avant de pouvoir prendre une décision au sujet des détails techniques de l'implantation interactive, il faut d'abord se définir une image précise de l'usager typique, auquel s'adresse le système. Dans le cas de EMME/2, l'usager typique est un planificateur de transport, qui satisfait aux exigences suivantes: il est familier avec les techniques principales de modélisation de réseaux de transport et il a acquis une connaissance approfondie de l'infrastructure de transport existante ainsi que des caractéristiques socio-économiques de la région qui fait l'objet de l'étude. On n'exige pas qu'il comprenne les détails mathématiques des modèles de prévision utilisés, ni qu'il soit un expert en informatique. Après le début d'une session de travail, il ne doit savoir effectuer que deux types de manipulation: entrer une ligne de texte au terminal et positionner le dispositif d'entrée graphique.

Une conséquence découlant directement de cette caractérisation de l'usager est la façon selon laquelle les responsabilités sont partagées entre l'usager et le système. Le système est responsable de tous les aspects informatiques. Il doit assurer que les actions de l'usager ne puissent jamais résulter en des erreurs informatiques, i.e. des situations qui causent une terminaison anormale du programme. L'usager, par contre, est responsable de l'aspect planification et de la logique de ses actions. C'est à lui de décider comment il va utiliser les possibilités offertes par EMME/2 et dans quel ordre. Tant qu'une opération demandée par l'usager est faisable (i.e. ne cause pas d'inconsistance informatique), le système l'acceptera. Il revient à l'usager de ne demander que des opérations qui

"ont du sens" dans son contexte.

Le système est donc "fool proof" dans le sens où il accepte toute entrée de l'usager, sans jamais aboutir à un problème informatique qui mette en danger l'intégrité de la banque de données ou qui cause un avortement du programme.

Par contre, le terme "fool proof" ne s'étend pas aux erreurs logiques commises par l'usager. Il est impossible que le système prévoie toutes les erreurs logiques qu'un usager pourrait faire. Par conséquent, le système n'essaie même pas de les identifier.

Le système EMME/2 peut être regardé comme un ensemble d'*outils*. La fonction de chaque outil est bien définie. Le système est responsable du bon fonctionnement de ses outils; mais c'est uniquement à l'usager de décider comment il va se servir de ces outils. Un usager novice va donc, de temps à autres, produire des résultats inutilisables, sans que le système ne soit intervenu. Des exemples typiques de cela sont des dessins vides ou trop denses. Un usager expérimenté, par contre, peut utiliser ces outils à leur pleine puissance, sans être limité par un système qui essaie de décider pour l'usager ce qui a du sens et ce qui n'en a pas.

Avoir une bonne idée de l'usager typique est essentiel pour pouvoir établir un format de communication interactive approprié. Faisons d'abord un tour d'horizon des différentes approches possibles pour une communication interactive entre l'usager et le système.

A un extrême, on demande à l'usager de prendre l'initiative et de définir exactement ce qu'il veut que le système fasse comme prochaine

opération. Ceci donne lieu à un système interactif controlé *par commandes*. L'usager écrit une commande qui décrit l'opération désirée, et le système répond en effectuant cette opération. La plupart des systèmes d'exploitation interactifs, incluant UNIX, communiquent avec l'usager de cette façon. L'avantage de l'approche par commandes est qu'elle permet une communication très compacte. Comme désavantage, on remarque que l'usager doit entrer des commandes qui peuvent devenir aussi complexes que les opérations désirées. L'usager doit donc apprendre tout un *langage de commandes* avant de pouvoir utiliser le système. Pour des systèmes peu complexes, cette approche a des avantages, surtout pour un usager expert qui connaît par coeur tout le langage de commande. Dès qu'un système interactif devient trop complexe pour qu'un usager puisse mémoriser toutes les commandes, ou que l'usager typique n'utilise pas assez souvent certaines commandes pour les savoir par coeur, l'interaction par commandes perd ses avantages. L'usager doit alors préparer à l'avance ses commandes à l'aide d'un manuel de référence - contredisant ainsi l'essentiel du mode de travail interactif.

A l'autre extrême, on a l'approche d'interaction *par menus*. Dans cette méthode, c'est le système qui mène l'usager, pas à pas, à travers toute la session de travail. A chaque point de décision, le système affiche un menu - une liste des actions possibles, parmi lesquelles l'usager doit choisir. L'usager ne doit rien mémoriser, puisque les choix possibles sont énumérés explicitement à chaque fois que l'usager est appelé à prendre une décision. Le choix est fait en entrant le numéro qui correspond à l'alternative désirée, ou, dans le cas d'une interaction graphique, en pointant le dispositif d'entrée graphique sur la réponse choisie. L'approche d'interaction par menu est idéale pour un usager

novice, puisqu'elle ne requiert presqu'aucun apprentissage. Pour un usager expérimenté, la même approche peut devenir fastidieuse: il connaît déjà par coeur les menus et ce n'est qu'une perte de temps de répéter le même menu détaillé pour la centième fois. Plus la vitesse de transmission est lente, plus ce point devient important. On remarque que la méthode d'interaction par menu ne s'adresse qu'au cas où l'usager se voit devant des choix discrets. Elle ne permet pas l'entrée de valeurs numériques, ni de chaînes de caractères. Afin d'accomplir cela, il faut ajouter un autre moyen de communication, d'habitude sous forme d'une question qui demande à l'usager d'entrer les valeurs nécessaires.

Les deux approches discutées ci-haut constituent les cas extrêmes. Evidemment, il est possible de les combiner pour obtenir des approches intermédiaires, qui incorporent des éléments des deux. L'approche choisie pour le système EMME/2 est un exemple d'une telle approche intermédiaire. Etant donnée la complexité du système - il comprend une cinquantaine de modules -, la méthode d'interaction par commandes pure est inapplicable. Or, la plupart des interactions se font sous forme de *dialogue*, une réalisation particulière d'un système interactif contrôlé par menus, augmenté d'un mécanisme pour l'entrée de valeurs numériques et textuelles. Ce dialogue consiste en une séquence alternante de questions posées par le système, et de réponses de l'usager. Les questions sont structurées d'une façon très stricte. On ne distingue que trois types de questions, comme on l'a vu dans la section 2.3. Deux types de questions appliquent le concept des menus: la question "YES/NO" est employée pour les alternatives dichotomiques, tandis que la question "SELECT" permet à l'usager de choisir une option parmi plusieurs. Le troisième type de question, la question "ENTER", sert à l'entrée de valeurs numériques ou de données textuelles.

A certains endroits-clefs, par contre, l'interaction par dialogue serait trop lourde. Puisqu'il s'agit de parties importantes du logiciel qui sont utilisées souvent par chaque usager, il est raisonnable de les implanter en utilisant des commandes au lieu d'un dialogue. Une interaction par commandes est utilisée

a) pour redéfinir l'environnement de travail. Les commandes peuvent être données au niveau du menu principal qui permet de choisir le prochain module à exécuter. Voir aussi la section III.4.1 du manuel de l'usager.

b) pour manipuler les sorties graphiques. L'utilisation d'un dialogue causerait la destruction partielle du dessin existant, que l'usager désire manipuler. Les *commandes graphiques*, comme on le verra dans la section 4.4, ne comportent qu'un seul caractère qui ne dérange pas le contenu de l'écran.

c) pour effectuer des modifications au contenu de la banque de données à l'aide d'une feuille de travail. Comme il deviendra évident dans la section 4.5, les feuilles de travail incorporent à la fois des menus, des commandes et des dialogues.

Règle générale, toute entrée que fait l'usager suit les conventions du *format libre*. Bien que le language FORTRAN '77 comprenne une lecture en format libre, celui-ci a été jugé inadéquat pour nos besoins, pour plusieurs raisons:

- Toutes les données textuelles doivent être délimitées par des guillemets. C'est une exigence fastidieuse et superflue dans notre contexte.

- Le format libre de FORTRAN '77 exige que l'usager entre *exactement* le nombre d'items demandé. Ceci rend difficile le traitement des items optionnels.

- Bien qu'une erreur à l'entrée puisse être attrapée par la clause ERR= de l'enoncé READ correspondant, il n'y a pas de façon standardisée d'identifier le type d'erreur qui est survenue; il est donc impossible d'envoyer des messages d'erreur précis à l'usager.

- Dans certains cas, il serait souhaitable de pouvoir traiter des mots-clefs pour identifier les items. Le format libre de FORTRAN 77 ne le permet pas.

C'est pour ces raisons qu'on a développé l'ensemble de sous-routines FREEFO, qui voit au traitement de l'entrée à format libre. Sans se perdre trop dans les détails, voici les caractéristiques principales du format libre utilisé dans EMME/2:

- Deux items consécutifs sont séparés par un espace blanc ( ), une virgule (,) ou des deux-points (:), optionnellement entourés d'espaces blancs supplémentaires. Un tel bloc est appelé un *séparateur*.

- Un "slash" (/) à la place d'un séparateur indique la fin logique de la ligne d'entrée. Ce qui suit est considéré comme commentaire.

- Un item qui n'est pas entré ne change pas la valeur de la variable correspondante. En général, une valeur non entrée a comme conséquence une action par défaut.

- FREEFO permet l'utilisation de mots-clefs pour identifier les items. Le mot-clef doit précéder directement l'item associé.

- Un **item numérique entier** peut être entré avec ou sans point décimal. La partie fractionnaire d'un item entier doit être nulle.

- Un **item numérique réel** peut être entré avec ou sans point décimal. La notation scientifique, avec exposants à la base dix, n'est pas actuellement implanté.

- Un **item alphanumérique** est normalement utilisé pour des identificateurs; il contient habituellement un "mot", i.e. une chaîne de caractères qui ne contient pas de séparateurs. L'item commence par le premier caractère non-blanc et finit dès qu'un séparateur est rencontré, ou, le cas échéant, quand le nombre maximal de caractères est atteint. Optionnellement, un item alphanumérique peut être délimité par des guillemets; dans ce cas, il peut contenir des séparateurs.

- Un **item de texte** est utilisé quand il s'agit d'une description, qui peut contenir des séparateurs. Il commence par le premier caractère non-blanc et se termine quand le nombre maximal de caractères est atteint ou à la fin de la ligne. Optionnellement, un item de texte peut être délimité par des guillemets; dans ce cas, il finit au deuxième guillemet.

Plus important et plus difficile que le choix du format de l'interaction est de bien choisir son *contenu*. Malheureusement, on se voit confronté à un compromis fondamental:

- Est-ce qu'on limite le dialogue au strict nécessaire, occasionnant une communication aussi compacte que possible?

- Ou, écrit-on les programmes de la façon la plus générale, donnant à l'usager une possibilité d'intervention pour tous les détails de l'opération.

Il est clair que les deux objectifs se contredisent. D'un coté, plus on simplifie le dialogue, plus le système doit prendre des décisions pour l'usager et utiliser des options "standard", qui ne coïncident pas nécessairement avec celles que l'usager désire. D'un autre coté, si plus de paramètres sont contrôlés par l'usager, ceci augmente forcément le nombre de questions à poser. Le dialogue devient donc inutilement lourd pour un usager ordinaire, qui ne veut que les options "standard".

Comment peut-on éviter ce dilemme et avoir en même temps un dialogue raisonnablement court pour un utilisateur typique *et* des programmes flexibles qui permettent à un usager avancé d'avoir un contrôle complet sur les options?

Dans EMME/2, ceci est réalisé à l'aide des *paramètres de module*. Le dialogue que l'usager doit entretenir pour accomplir une certaine opération ne contient que les questions essentielles pour définir les paramètres de base de cette opération. Cependant, chaque module contient aussi d'autres paramètres - les paramètres de module - qui influencent les résultats de

son opération. Ces paramètres déterminent en détail *comment* l'opération doit être effectuée, comme par exemple, les paramètres de la représentation graphique, les valeurs de seuil, l'inclusion/suppression des éléments vides, etc. Au début de l'exécution d'un module, ses paramètres de module sont initialisés à des valeurs par défaut. Un usager qui ne veut qu'utiliser ces valeurs par défaut n'a rien à faire. Si, par contre, l'usager désire changer les valeurs de ces paramètres, il peut faire cela en choisissant l'option correspondante dans la première question "SELECT" du module. Dans ce cas, il doit parcourir une séquence de dialogue qui lui permet de définir tous les paramètres de ce module. Les nouvelles valeurs ainsi définies peuvent avoir un caractère temporaire (elles sont oubliées en quittant le module) ou elles peuvent devenir les nouvelles valeurs par défaut. Le concept des paramètres de module permet donc à l'usager d'avoir un contrôle très fin sur les détails du module quand il le désire, sans allonger inutilement les dialogues parcourus dans un traitement ordinaire.

Tandis que la sphère d'influence d'un paramètre de module est limitée au module même, EMME/2 permet aussi un certain contrôle global de l'interaction entre l'usager et le système. Ceci est réalisé à l'aide d'un ensemble de *commutateurs* (en anglais "switch"), qui permettent à l'usager d'"allumer" ou d'"éteindre" certaines séquences de dialogue à travers tout le système. Par exemple, un usager qui est branché sur un terminal à basse vitesse perdrait beaucoup de temps chaque fois que le menu principal - la liste de modules disponibles - est affiché. En éteignant le commutateur correspondant, l'affichage du menu principal est supprimé, sauf si demandé explicitement. Ou encore, un usager, après avoir établi une banque de données, peut vouloir protéger les valeurs par défaut des paramètres de module contre une modification par inadvertance; il n'a qu'à supprimer la

question qui permet de sauver les valeurs courantes comme nouvelles valeurs par défaut. Une description de tous les commutateurs se trouve dans la section III.4.1 du manuel de l'usager.

Enfin, un dernier point, néanmoins important, est celui de la langue utilisée pour la communication entre l'usager et le système. Pour le développement de EMME/2, nous avons choisi l'anglais, plutôt que le français, et ce pour plusieurs raisons:

- La grande majorité de la littérature pertinente dans ce domaine est publiée en anglais. Par conséquent, la terminologie anglaise est plus standardisée que la française.

- L'intérêt pour un outil interactif-graphique pour la planification des transports urbains comme EMME/2 n'est pas limité aux villes du Québec, mais s'étend au reste du Canada et même à l'étranger. L'anglais maximise l'accessibilité au logiciel par les différents usagers potentiels.

- En choisissant l'anglais comme la langue de communication, on évite le problème de la représentation informatique des accents français, un problème qui jusqu'à maintenant, n'a malheureusement pas encore de solution standardisée.

3. Structures des données

Dans la section 2.4 de ce chapitre, nous avons élaboré les concepts fondamentaux de la banque de données de EMME/2, sans s'occuper des détails de son contenu logique. Dans cette section, nous présenterons les données nécessaires pour le fonctionnement de EMME/2, et nous étudierons leur représentation et leur organisation dans la banque de données.

La figure III-3-1 donne une vue d'ensemble du contenu d'une banque de données de EMME/2.

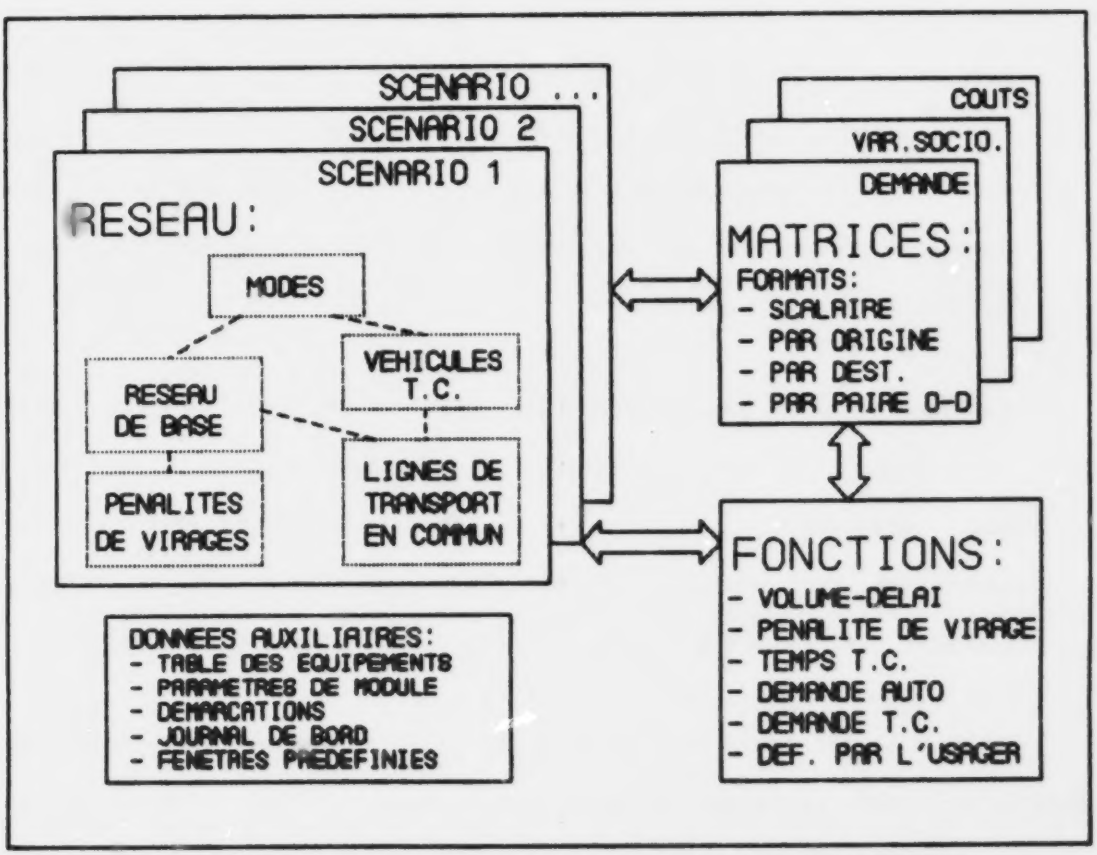

FIGURE III-3-1: Contenu de la banque de données de EMME/2.

Les données associées avec EMME/2 se regroupent en trois ensembles principaux:

1) **Les données de réseau:** Cet ensemble de données fournit la description complète d'une *infrastructure de transport* dans une région, ainsi que ses résultats associés (temps et volumes), après qu'une affectation ait été faite. Une telle description de réseau est appelée un *scénario*. Une banque peut contenir plusieurs scénarios.

2) **Les données matricielles:** Cet ensemble regroupe toutes les données qui sont associées avec les *zones origines et destinations*. On y trouve les matrices de demande, les données socio-économiques caractérisant les zones ou les interactions entre les zones, et des données résultantes qui décrivent le niveau de service. Bien que ces données soient très différentes dans leur signification, elles sont identiques dans leur représentation. Le système les traite comme des *matrices*, de quatre formats différents.

3) **Les fonctions:** Un aspect unique de EMME/2 est le fait que les relations fonctionnelles utilisées dans les modèles de prédiction ne sont pas programmées d'une façon fixe, mais constituent des données, spécifiées par l'usager et stockées dans la banque de données. L'usager les définit à l'aide d'*expressions algébriques et logiques*.

En plus de ces trois parties principales, la banque de données de EMME/2 contient certaines *données auxiliaires*. Elles constituent, en général, des aides à l'usager, lui permettant une utilisation plus facile du système.

Les quatres sections suivantes explorent plus en détails les données associées avec chacune des parties mentionnées plus haut.

3.1 Le réseau

Dans EMME/2, le terme "réseau" est utilisé dans un sens très général; il inclut tous les éléments nécessaires à la description de l'*infrastructure de transport*.

Plus précisément, dans EMME/2 un réseau comprend six types d'éléments. Les modes caractérisent les types de services de transport offerts. Les noeuds et les liens forment un graphe, appelé le *réseau de base*, qui décrit toutes les connexions spatiales possibles. Pour le trafic privé, certains virages peuvent être interdits ou pénalisés. Pour le transport en commun, on spécifie les types de véhicule utilisés, ainsi que les itinéraires des lignes qui sont opérées.

Avant de porter notre attention à l'implantation de ces six types d'éléments, il est nécessaire de les définir d'une façon plus détaillée:

MODES: On distingue trois types de mode de transport. Le *mode de transport privé* (ou plus simplement, le *mode auto*) est normalement utilisé pour les déplacements par voiture privée. Les *modes de transport en commun* permettent les déplacements sur des lignes de transport en commun à itinéraire fixe. Finalement, les *modes auxiliaires* sont utilisés pour tout déplacement par transport en commun qui n'emploie pas de lignes, mais se fait lien par lien. L'utilité de ces modes auxiliaires réside surtout dans la

modélisation de l'accès, l'egrès et les transferts entre les lignes.

NOEUDS: Un noeud désigne un endroit précis dans la région géographique considérée. Sa position est donnée par une paire de coordonnées X-Y.

Un noeud représentant le centre d'une zone origine/destination est appelé un *centroïde*. Tout voyage commence à un centroïde et se termine à un centroïde. Un centroïde ne peut pas constituer un noeud intermédiaire dans un voyage.

Un noeud représentant un carrefour avec des virages interdits ou pénalisés est appelé une *intersection*.

LIENS: Un lien est une connexion directionnelle reliant deux noeuds. Les boucles, i.e. les liens reliant un noeud à lui-même, et les liens parallèles, reliant les mêmes deux noeuds dans la même direction, ne sont pas admis.

Pour chaque lien, un sous-ensemble non-vide de modes désigne les types de service possibles sur ce lien.

Un lien relié à un centroïde est appelé un *connecteur*.

VIRAGES: Un virage désigne un mouvement tournant à un noeud intersection. Un virage peut être *permis* (aucun délai), *pénalisé* (délai selon une fonction de pénalité) ou *interdit*. Les virages servent à modéliser les caractéristiques d'un carrefour; ils ne sont définis que pour le mode auto.

Par défaut, tous les virages sont permis, sauf les *virages en U*,

qui sont interdits.

VEHICULES: Le matériel roulant utilisé pour les lignes de transport en commun est classifié par *types de véhicule*. Pour chaque type, on spécifie la flotte, la capacité, ainsi que les coefficients de coût et de consommation d'énergie. Un type de véhicule ne peut servir qu'à un mode.

LIGNES: Une ligne de transport en commun est définie comme un service régulier sur un itinéraire fixe. L'itinéraire est une suite de segments, décrivant une séquence de liens sur le réseau de base. Chaque segment correspond à un seul lien. Une ligne peut avoir un ou deux *terminus*, dont le dernier correspond toujours avec le dernier noeud de l'itinéraire. Le premier et le dernier noeud d'une ligne ne sont pas nécessairement identiques.

La figure III-3-2 montre les six types d'élément et donne les attributs principaux de chacun, ainsi que les résultats qui sont calculés par les modules d'affectation. L'existence de pointeurs d'un type d'élément vers un autre est indiquée par des flèches.

Portons maintenant notre attention à l'organisation interne de ces éléments du réseau. En général, l'organisation utilisée dans la banque de données coïncide avec celle utilisée en mémoire centrale pendant l'exécution des modules, à l'exception des attributs non-nécessaires. Pour clarifier les concepts, nous utiliserons les même noms de variables que dans le programme source de EMME/2, documentés dans l'appendice C du manuel de l'usager. Tous les noms de variables sont donnés en majuscules. Pour

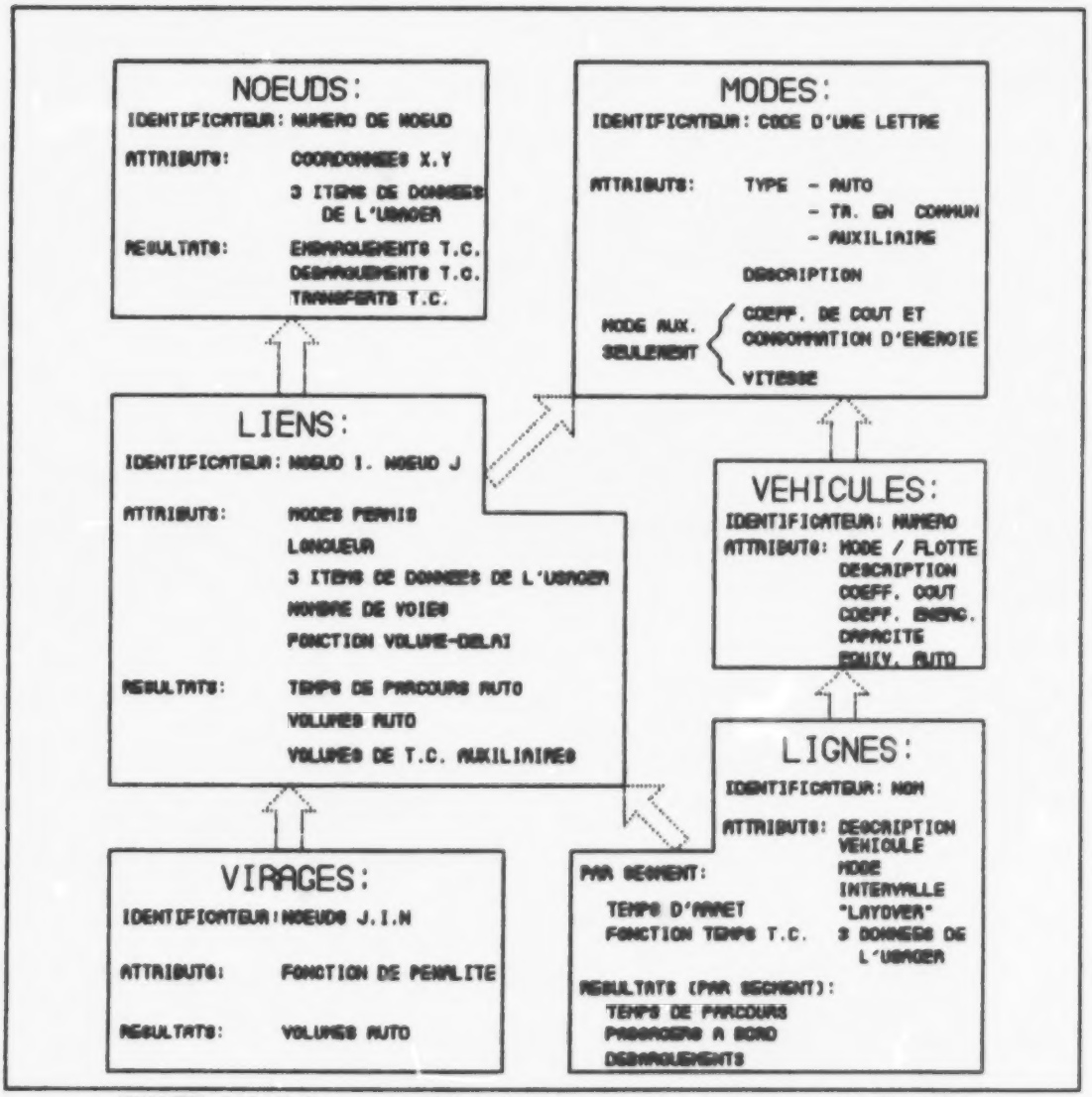

FIGURE III-3-2: Les éléments de réseau et leurs attributs.

les variables vectorielles, les indices utilisés suivent la convention suivante:

M	mode
I,J	noeud
L	lien
K	virage
IV	véhicule de transport en commun
IL	ligne de transport en commun
IT	segment de ligne

Les séquences de programme présentées dans cette section sont écrites en

FORTRAN '77.

Les modes de transport sont représentés sous forme d'une table simple avec indices M=1,2,...,MMODE. Le mode M=1 est réservé pour le mode auto. Chaque mode est identifié par *une lettre*. Contrairement aux autres tables d'éléments, la table des modes n'est pas triée selon les identificateurs des éléments, mais est gérée d'une façon statique. Deux raisons justifient ce choix. Primo, la table des modes est très courte, une application typique en ayant de 3 à 5. Secundo, un réarrangement des indices des modes impliquerait des changements majeurs dans la table des liens. Ce dernier point découle directement de la façon dont les sous-ensembles de modes sont traités, comme par exemple, le sous-ensemble des modes permis sur un lien. Etant donné le nombre relativement petit de modes différents, un sous-ensemble de modes peut aisément être représenté sous la forme d'un *patron de bits* (en anglais, "bit pattern"), le bit le plus faible correspondant au mode M=1. Un bit de valeur 1 indique l'appartenance du mode correspondant au sous-ensemble. Cette méthode donne lieu, en même temps, à une représentation très compacte et à un traitement extrêmement efficace. Les opérations de base sur les sous-ensembles, l'intersection, l'union et les tests d'appartenance, se réduisent à des opérations logiques *ET* et *OU* sur les mots correspondants. Pour un ordinateur à mots de 16 bits, le maximum de modes est donc de 15, et pour un ordinateur à mots de 32 bits, il est de 31 ... ce qui est amplement suffisant pour l'application la plus exigeante. On remarque que le bit de signe du mot n'est pas utilisé dans le patron de bits. Ceci permet d'utiliser des nombres entiers positifs pour représenter un sous-ensemble de modes.

Les noeuds sont stockés dans une table qui est divisée en deux

parties. La première partie contient les *centroïdes* et la deuxième, les *noeuds réguliers*. Chaque partie est triée en ordre croissant des identificateurs des noeuds, i.e. des *numéros de noeuds*. La table peut contenir un maximum de MNODE noeuds, incluant un maximum de MCENT centroïdes et de (MNODE-MCENT) noeuds réguliers. La table des noeuds contient donc, en général, un "trou" entre le dernier centroïde, qui porte l'indice NCENT, et le premier noeud régulier, qui est situé à (MCENT+1). La variable NNODE indique la position du dernier noeud régulier, ou, s'il n'en existe aucun, a la valeur MCENT. Le tableau III-3-i illustre le contenu de la table des noeuds.

Indice	1 2 3 . . . NCENT . . . MCENT	NCENT+1 . . . . . NNODE . . . MNODE
Contenu	(------centroïdes-----) (non-utilisés)	(-----noeuds réguliers-----) (non-utilisés)

TABLEAU III-3-i: La table des noeuds.

Cette organisation de la table des noeuds permet une correspondance directe entre les centroïdes et les zones origines/destinations qui sont utilisées dans la définition des matrices, comme on le verra dans la section suivante.

La table des liens est triée selon les identificateurs des liens. Puisqu'un lien est identifié par les identificateurs de son noeud initial I et de son noeud terminal J, le réseau de base est organisé en ordre *vers l'avant* (en anglais, "forward star"). Ceci implique que tous les liens qui sortent du même noeud sont adjacents dans la table des liens. Etant stockés dans cet ordre, il n'est pas nécessaire de garder explicitement un pointeur vers le noeud initial de chaque lien. Il suffit de connaître pour

chaque noeud I, quel est le premier lien dans la table qui sort d'un noeud $\geq$ I. Ce pointeur est appelé IPT(I). Si on dénote par JNODE(L) le pointeur vers le noeud terminal J du lien L, un parcours du réseau est fait par la séquence simple

```
      DO 20  I=1,NNODE
         DO 10  L=IPT(I),IPT(I+1)-1
         J=JNODE(L)
            . . .
            <traitement du lien L de I à J>
            . . .
   10    CONTINUE
   20 CONTINUE
```

Il y a deux détails importants à noter. Premièrement, cette séquence fonctionne même avec un "trou" dans la table des noeuds, puisque dans ce cas, l'intervalle [IPT(I),IPT(I+1)-1] est vide. Deuxièmement, pour I=NNODE, la séquence fait appel à IPT(NNODE+1), qui doit toujours pointer vers le premier indice de lien non-utilisé.

Pour certaines tâches, comme par exemple l'algorithme d'affectation du transport en commun décrit dans le chapitre II, l'organisation du réseau vers l'avant n'est pas désirable, puisqu'il nous faut accéder efficacement à tous les liens qui entrent dans un noeud particulier. Dans ces cas, il faudrait organiser le réseau de base *vers l'arrière* (en anglais, "backward star"). Cette organisation revient à une organisation vers l'avant du réseau inversé, i.e. du réseau obtenu en changeant la direction de chaque lien. L'inversion du réseau, i.e. la construction des vecteurs JPT et INODE à partir des vecteurs IPT et JNODE, est d'une complexité de calcul linéaire, comme le montre la séquence suivante:

```
      DO 10  J=1,NNODE+1
         JPT(J)=0
   10 CONTINUE

      DO 20  I=1,NNODE
         DO 15  L=IPT(I),IPT(I+1)-1
            J=JNODE(L)
            JPT(J)=JPT(J)+1
   15    CONTINUE
   20 CONTINUE

      JPT(1)=JPT(1)+1
      DO 30  J=1,NNODE
         JPT(J+1)=JPT(J)+JPT(J+1)
   30 CONTINUE

      DO 40  I=NNODE,1,-1
         DO 35  L=IPT(I+1)-1,IPT(I),-1
            J=JNODE(L)
            JPT(J)=JPT(J)-1
            INODE(JPT(J))=I
   35    CONTINUE
   40 CONTINUE
```

En effet, il est possible de montrer que cette méthode est une variante de l'algorithme de tri par comptage de distribution (Knuth 1973), qui est bien connu pour sa complexité linéaire. L'inversion des limites de la dernière boucle nous assure que le réseau inversé est ordonné selon les numéros de noeuds J-I. Puisque la conversion d'un réseau représenté vers l'avant en représentation vers l'arrière est si efficace, il est inutile de stocker le réseau inversé dans la banque de données.

Comparée à la structure du réseau de base, l'organisation des virages est assez complexe. Au niveau de leur définition, ils sont stockés dans une forme condensée qui ne contient que les virages pénalisés ou interdits. Un virage est identifié par un triplet de noeuds (α,β,γ), où α et le noeud d'intersection; le virage correspond à un mouvement venant du noeud β et allant vers le noeud γ, impliquant l'existence des deux liens (β,α) et (α,γ). Les virages ainsi définis sont stockés, dans l'ordre des identifi-

cateurs croissants, dans une structure de la forme

$$\{\Lambda_\alpha,\ \alpha,\ \{-\beta,\ \{\gamma,\ p_{\alpha\beta\gamma},\ \}\ \}\ \}\ 0$$

où α, β et γ sont les identificateurs des noeuds correspondants et Λ_α est la longueur du bloc de données associé avec α. Le code de pénalité $p_{\alpha\beta\gamma}$ est l'indice d'une fonction de pénalité, pour un virage pénalisé, ou 0, pour un virage interdit. Les accolades {} indiquent que le bloc qu'elles entourent est répété autant de fois que nécessaire. Une valeur $\Lambda_\alpha=0$ indique la fin de la structure. Cette structure minimise l'espace requis pour stocker la définition des virages pénalisés et interdits. Puisqu'elle contient des identificateurs des noeuds, et non pas des pointeurs vers les noeuds, la mise-à-jour de cette structure, suite à une modification du réseau de base, est très simple.

Bien que cette structure soit idéale pour stocker la définition des intersections et virages, elle est très lourde lorsqu'utilisée dans les algorithmes. De plus, elle ne peut pas être utilisée pour le stockage des résultats de l'affectation (i.e. le volumes sur les virages), parce que ceux-ci sont calculés pour tous les virages de toutes les intersections. C'est pour cette raison que, lors de la préparation d'une affectation-auto, une structure différente est bâtie, qui est mieux adaptée aux algorithmes et qui peut contenir les résultats générés. Puisqu'à cet instant-là, il ne peut plus y avoir des changements du réseau, il est possible de combiner les structures du réseau de base avec celles des virages aux intersections. Ceci est réalisé en changeant légèrement la définition du vecteur JNODE. Dans le cas des liens qui n'aboutissent pas à une intersection, la forme modifiée du vecteur JNODE (appelée JNODEP afin de la distinguer du vecteur original) contient encore le pointeur vers le noeud terminal du lien. Si,

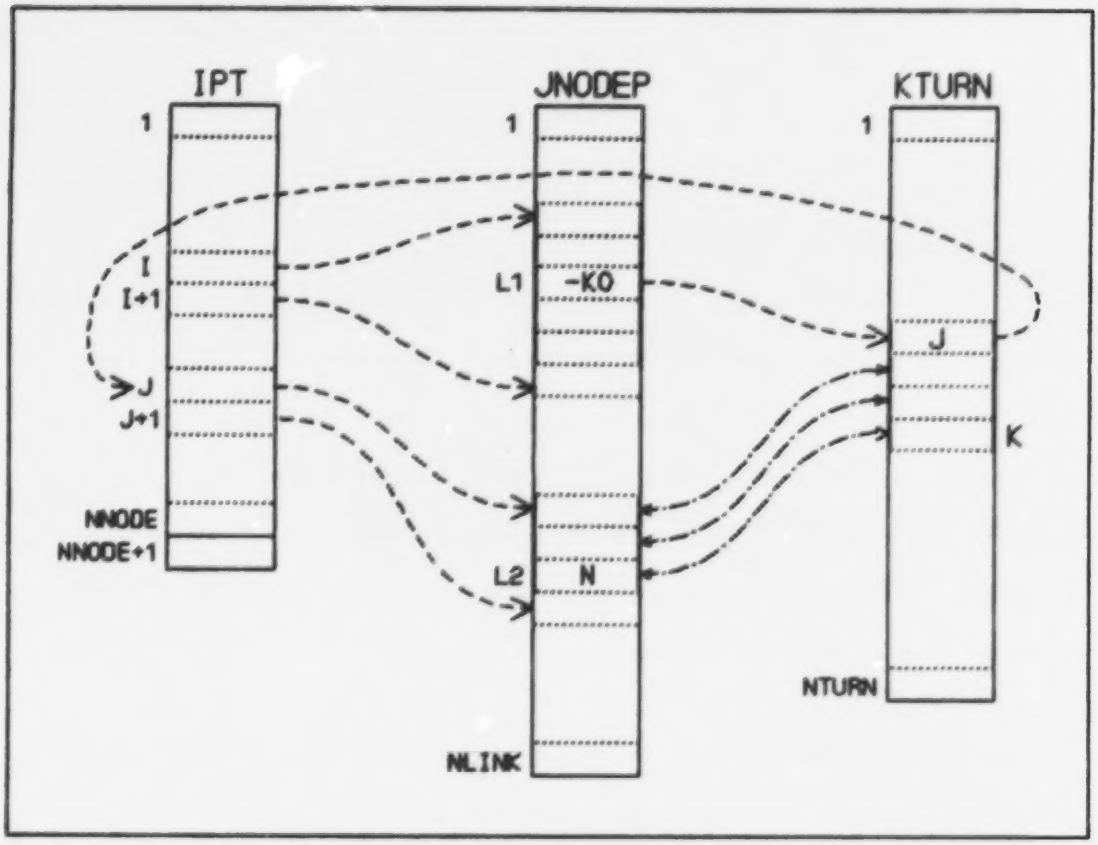

FIGURE III-3-3: La structure de données associée aux virages.

par contre, le noeud terminal du lien L1 est une intersection, JNODEP(L1) contient un pointeur vers un élément d'un nouveau vecteur, KTURN, qui contient l'information sur les intersections. Le signe de JNODEP(L1) est utilisé pour identifier ce dernier cas. Soit KO l'indice de KTURN contenu dans JNODEP(L1). KTURN(KO) contient alors J, le noeud terminal du lien L1. KTURN(K), où i=KO+1,...,KO+IPT(J+1)-IPT(J), contient le code de pénalité du virage du lien L1 à L2=IPT(j)+i-1. L'avantage de cette méthode est qu'on utilise deux fois l'information contenue dans IPT et JNODE: une fois pour le lien incident L1 et l'autre fois pour le lien sortant L2. Les attributs des virages peuvent facilement être stockés dans des vecteurs qui sont parallèles au vecteur KTURN.

La figure III-3-3 illustre l'organisation interne des intersections. A titre d'exemple de l'utilisation de cette structure, voici une séquence d'instructions qui permet de parcourir tous les virages.

```
      DO 100  I=1,NNODE
         DO  90  L1=IPT(I),IPT(I+1)-1
            IF(JNODEP(L1).GE.0)GOTO 100
            K=-JNODEP(L1)
            J=KTURN(K)
            DO 80  L2=IPT(J),IPT(J+1)-1
               K=K+1
               N=JNODEP(L2)
               IF(N.LT.0)N=KTURN(-N)
                   . . .
                   traitement du virage (J,I,N)
                   reliant les liens L1 et L2 avec
                   code de penalité KTURN(K)
                   . . .
 80         CONTINUE
 90      CONTINUE
100  CONTINUE
```

Les types de véhicule sont organisés en une table simple, similaire à celle de modes. Comme identificateurs des types de véhicule, on utilise directement leurs indices dans la table, i.e. des nombres entre 1 et MVEH. Pour chaque véhicule, la table contient les caractéristiques d'opération ainsi qu'un pointeur qui indique le mode de transport en commun auquel appartient ce type de véhicule.

Les lignes de transport en commun sont stockées dans la banque de données à l'aide de deux tables, la *table des lignes*, qui contient les informations s'appliquant à toute la ligne, et la *table des segments*, qui contient la définition des itinéraires. L'identificateur d'une ligne est *un nom de 6 caractères ou moins.* Dans la table des lignes, les lignes apparaissent dans l'ordre alphabétique des noms aux adresses 1 à NLINE. Pour chaque ligne IL, la table des lignes contient le nombre de segments

par ligne LINLST(IL), ainsi qu'un pointeur vers la table des segments, LINPLS(IL), qui indique la position du premier segment de cette ligne. Les segments de la ligne IL se trouvent donc aux adresses LS = LINPLS(IL), ..., (LINPLS(IL)+LINLST(IL)-1) de la table des segments.

Quelle information faut-il conserver avec chaque segment de ligne pour permettre la reconstruction de l'itinéraire sur le réseau de base? Puisque chaque segment correspond à un lien, la solution la plus facile serait donc de stocker, avec chaque segment, un pointeur vers le lien correspondant du réseau de base. Cette approche a deux désavantages: premièrement, pour trouver la séquence des noeuds parcourus par l'itinéraire de la ligne, il faut garder en mémoire un copie complète du vecteur JNODE; deuxièmement, chaque ajout ou élimination d'un lien du réseau de base entraînerait des changements majeurs de ces pointeurs. Une autre solution possible consiste à stocker, avec chaque segment, un pointeur vers le noeud initial du segment. Dans ce cas, il est trivial de reconstruire la séquence de noeuds, mais, pour pouvoir retrouver la séquence de liens, il ne faut pas seulement avoir en mémoire la structure complète du réseau de base (IPT et JNODE), mais il faut aussi faire une *recherche* pour retrouver le lien en question parmi les liens sortant du noeud initial du segment. Comme on le voit, ni l'une, ni l'autre des deux approches ne satisfont tous les besoins; de plus, pour des raisons d'espace, on ne peut pas se permettre d'utiliser pour cette information plus d'un mot par segment.

La solution retenue pour EMME/2 tient compte de cette contrainte, tout en donnant accès en même temps aux noeuds et aux liens. Ceci est accompli par le vecteur ILINK qui contient, d'une façon compactée, un pointeur vers le noeud initial du segment et la position relative du lien parmi les liens

sortant du noeud initial. Supposons que I corresponde au noeud initial du segment LS et que LDELTA soit la position relative du lien correspondant. On n'a alors besoin que du vecteur IPT pour calculer l'indice du lien correspondant par L=IPT(I)+LDELTA. Notons que la position relative du lien, LDELTA, ne peut jamais dépasser le demi-degré extérieur maximal des noeuds, qui dépasse rarement 5 ou 6 dans un réseau de transport. Même avec un ordinateur à mots de 16 bits, cette approche permet de traiter jusqu'à 4095 noeuds et un demi-degré extérieur maximal de 16 ... ce qui est amplement suffisant pour la grande majorité des applications.

Avec cette structure, le noeud terminal J d'un segment LS peut être obtenu soit par JNODE(L), si ce vecteur se trouve en mémoire, soit par le noeud initial du segment suivant LS+1. Pour permettre de trouver le dernier noeud de l'itinéraire par la deuxième méthode, il est nécessaire d'ajouter un segment additionnel à la fin de l'itinéraire. Ce segment "bidon", qui se trouve à l'adresse LINPLS(IL)+LINLST(IL), ne contient qu'un pointeur vers le noeud final de l'itinéraire.

Afin d'illustrer cette structure, voici une séquence d'instructions qui calcule la longueur des itinéraires, LINLEN, pour toutes les lignes, à partir de la longueur des liens LEN. On suppose que les vecteurs IPT, LEN et ILINK sont présents en mémoire centrale et que les fonctions INOD() et LDELTA() sont utilisées pour décomposer ILINK.

```
      DO 20  IL=1,NLINE
         LINLEN(IL)=0
         LS1=LINPLS(IL)
         LS2=LS1+LINLST(IL)-1
         DO 10  LS=LS1,LS2
            I=INOD(ILINK(LS))
            L=IPT(I)+LDELTA(ILINK(LS))
            LINLEN(IL)=LINLEN(IL)+LEN(L)
   10    CONTINUE
   20 CONTINUE
```

3.2 Les matrices

Dans le système EMME/2, toutes les données associées aux zones origines / destinations sont traitées d'une façon unifiée, sous forme de matrices. Le système ne sait même pas quel genre de données (nombre de voyages, niveau de service, caractéristiques socio-économiques, etc.) est contenu dans telle matrice. La gestion du contenu logique des matrices est laissée à l'usager. Cette unification facilite en même temps l'implantation et l'interface entre l'usager et le système. De plus, elle permet la réalisation d'outils de gestion très généraux, comme l'utilitaire de génération d'histogrammes et la calculatrice de matrices que nous verrons dans la section 5.

Dans le contexte de EMME/2, une matrice est un ensemble de données numériques qui dépendent des zones origines et / ou destinations. Quatre formats de matrice existent:

MF Une *matrice origine - destination*, aussi appelée *matrice pleine*, est un ensemble de données contenant une valeur pour chaque paire origine - destination (O-D).

MO Une *matrice origine* est un vecteur de données contenant une valeur pour chaque zone origine.

MD Une *matrice destination* est un vecteur de données contenant une valeur pour chaque zone destination.

MS Finalement, une *matrice scalaire* consiste en une seule valeur numérique.

La figure III-3-4 illustre ces différents formats de matrices.

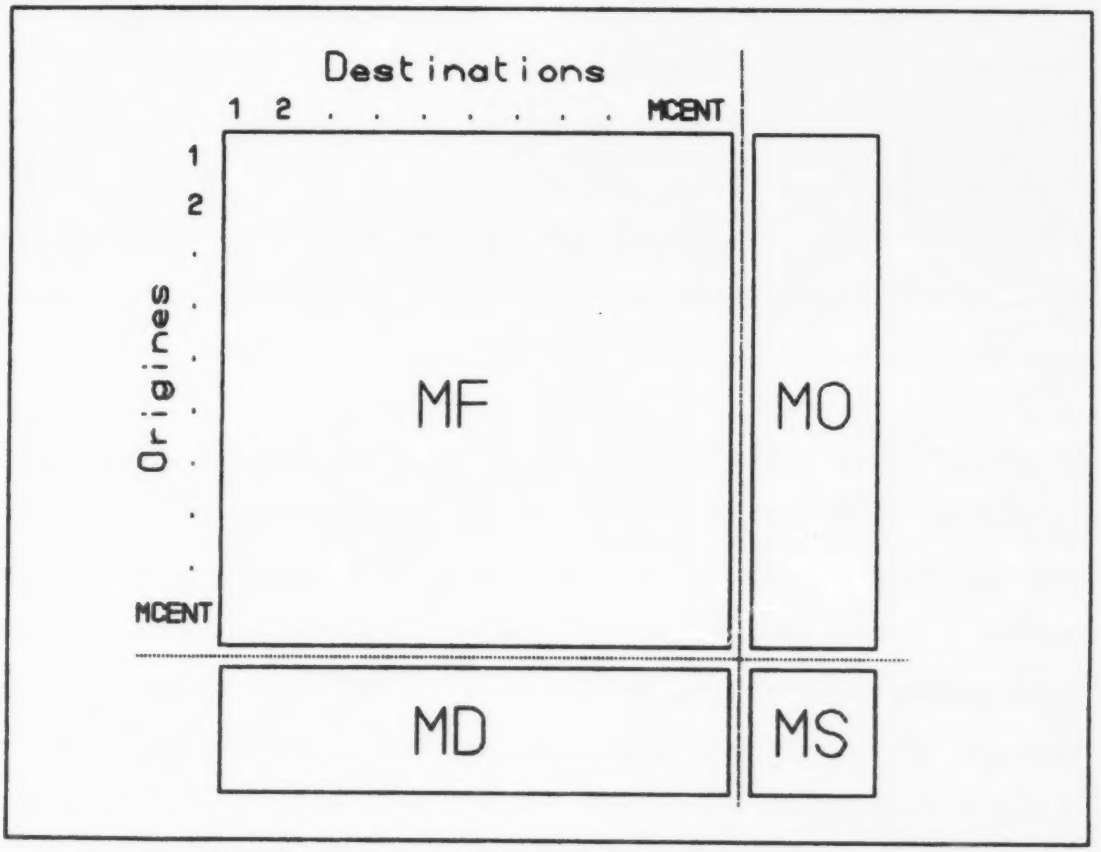

FIGURE III-3-4: Les quatre formats de matrice.

Comme on le voit, le terme "matrice" a une signification très spécifique dans EMME/2. En effet, l'existence de matrices *scalaires*

pourrait sembler paradoxale à première vue. Mais comme nous l'expliquerons plus loin, elle nous est très utile.

Les quatre formats permettent un stockage adéquat des différentes données associées aux zones. Dans une étude du trafic routier à l'heure de pointe du matin, par exemple, la matrice de demande serait stockée en format MF, les taux de possession d'auto en format MO et les coûts de stationnement en format MD.

Pendant l'exécution des modules de EMME/2, l'accès à ces matrices est effectué en utilisant une conversion automatique entre les divers formats. Cela veut dire qu'un module peut faire accès à une matrice dans un format différent du format physique de la matrice dans la banque de données. Les extensions nécessaires sont automatiquement faites par les routines d'écriture et de lecture de matrices. Ceci permet de formuler les modèles de la façon la plus générale, sans obliger l'usager à entrer ses donnees dans le format général.

Examinons par exemple les taux d'occupation des autos, i.e. les coefficients qui permettent de convertir le nombre de personnes qui voyagent en nombre de voitures. Selon l'application concrète et les sources de données disponibles, ces taux peuvent être spécifiés pour chaque paire O-D, comme valeurs moyennes par origine, comme valeurs moyennes par destination ou même par une moyenne globale pour tout le réseau. Selon le cas, l'usager peut stocker cette information dans une matrice de format MF, MO, MD ou MS. Dans le modèle d'affectation, par contre, ces taux d'occupation sont définis au niveau de chaque paire O-D, le cas le plus général. Le module d'affectation demande de lire cette matrice en format MF, sans connaître

dans quel format la matrice est physiquement stockée.

Le tableau III-3-2 donne un résumé des formats d'accès permis pour lire et écrire une matrice d'un format physique donné.

Format physique	Formats de lecture permis	Formats d'écriture permis
MF	MF -- -- --	MF MO MD MS
MO	MF MO -- --	-- MO -- MS
MD	MF -- MD --	-- -- MD MS
MS	MF MO MD MS	-- -- -- MS

TABLEAU III-3-2: Conversation automatique entre les formats de matrice.

Au moment de la création de la banque de données, l'usager spécifie le nombre de matrices de chaque format, contenues dans la banque. Puisque la plupart des données associées aux zones ne sont pas limitées à des valeurs entières, toutes les matrices sont représentées à l'aide des nombres à point flottant.

Pour le format de matrice MF, l'organisation interne peut être faite *par rangées* (accès efficace à toutes les valeurs associées avec la même origine) ou *par colonnes* (accès efficace à toutes les valeurs associées avec la même destination). Par défaut, l'organisation par rangées est utilisée. De façon interne, l'organisation par colonnes est employée pour l'affectation de transport en commun, puisqu'elle est basée sur les destinations (voir chapitre II). L'usager peut aussi forcer l'organisation par colonne d'une matrice, si, par exemple, il prévoit beaucoup d'accès par destinations à cette matrice.

Il est à noter que toutes les matrices origine - destination (format MF) sont stockées dans la banque de données telles quelles, sans appliquer une quelconque technique de compression pour les matrices creuses. Chaque matrice origine - destination utilise donc un espace-disque de $MCENT^2$ mots. Pour des matrices de demande de très basse densité, ceci peut apparaître une méthode inefficace en terme d'espace-disque. Il y a, par contre, plusieurs raisons importantes qui contre-indiquent l'utilisation de telles techniques:

- L'accès de lecture devient considérablement plus lent.

- Chaque accès d'écriture entraîne, en général, des modifications coûteuses de la structure des pointeurs.

- Pour une matrice pleine, qui n'a aucun élément nul, on utilise plusieurs fois l'espace requis pour un stockage simple.

- Dans une application typique, peu de matrices ont une densité assez basse pour profiter des techniques de compression.

Puisqu'il existe une correspondance bi-univoque entre les zones origines / destinations et les centroïdes, les zones sont identifiées par les identificateurs de noeud des centroïdes.

Pour permettre de travailler facilement avec des sous-ensembles de zones, des sous-matrices quelconques et des zones agrégées, l'usager peut définir des groupes de zones. Une banque de données peut contenir jusqu'à 26 ensembles de groupes, qui sont dénotés de "A" à "Z". Chacun de ces ensembles constitue une partition des zones en groupes, numérotés de 0 à

999. Le concept de groupes de zones permet à l'usager de travailler aisément avec des regroupements géographiques ou logiques de zones. Dans une application typique, un ensemble de groupes est utilisé, par exemple, pour partitionner les zones en communautés, un autre les divise en catégories "ville" et "campagne", tandis qu'un troisième ensemble les regroupe en zones agrégées.

3.3 Les fonctions

La plupart des modèles de prévision font appel à certaines relations fonctionnelles, reliant les diverses composantes du modèle. L'exemple par excellence est celui des *fonctions volume-délai*, qui représentent la relation entre le volume de trafic sur un lien et le temps nécessaire pour le franchir. Ces fonctions sont utilisées dans les modèles d'affectation routière, pour tenir compte des effets de congestion. Les modèles exigent seulement que ces fonctions existent et qu'elles satisfassent certaines conditions (positivité, monotonicité), mais laissent libre le choix de la forme fonctionnelle. Ces fonctions constituent donc une composante de la spécification du problème et, comme telles, devraient faire partie des *données du problème*.

Si on examine les logiciels de planification de transport existants, on constate, malheureusement, que ceci n'est pas le cas. Dans beaucoup de systèmes, la forme fonctionnelle des relations est incorporée d'une façon fixe au niveau de la programmation - ne laissant à l'usager que la spécification des valeurs des coefficients. Pour revenir à l'exemple, un choix fréquent pour les fonctions volume-délai est la forme BPR (Bureau of

Public Roads, 1963)

$$s_a(v_a) = t_a \left\{ 1 + \left\{ \frac{v_a}{c_a} \right\}^{\gamma_a} \right\} ,$$

où l'usager doit se contenter de spécifier trois coefficients: le temps non-congestionné t_a, la capacité du lien c_a, et l'exposant γ_a. D'autres systèmes (voir par exemple, Florian et al. [1977]) permettent, au moins en principe, l'utilisation de formes fonctionnelles quelconques. La définition des fonctions, par contre, ne se trouve pas dans la partie des données du problème, mais est faite sous forme de "sous-routine fournie par l'usager". Dans ces systèmes, l'usager est obligé d'écrire un nouveau sous-programme et de recompiler le système au complet chaque fois qu'il veut changer une fonction. Ceci n'est pas seulement une tâche longue et fastidieuse, mais elle requiert aussi de solides connaissances informatiques de la part de l'usager.

A notre connaissance, EMME/2 est le premier système de planification de transport qui permette à l'usager de spécifier les fonctions utilisées dans les modèles comme une partie des données du problème. Dans les pages suivantes, nous verrons de plus près comment les fonctions sont implantées

Présentement, il y a six *classes de fonctions* définies dans EMME/2, chacune dénotée par un code de deux lettres:

FD: fonctions volume-délai

FP: fonctions de pénalité de virage

FT: fonctions de temps de parcours du transport en commun

FA: fonctions de demande-auto

FB: fonctions de demande-transport en commun

FU: fonctions générales spécifiées par l'usager

Les premières cinq classes de fonctions correspondent aux relations fonctionnelles qui sont utilisées dans les modèles de prévision. La sixième classe n'est pas utilisée en soi dans le système. Elle a été ajoutée pour permettre à un usager de se servir des outils de gestion de fonctions offerts par EMME/2 pour des fonctions arbitraires, même si elles n'ont aucun rapport avec les problèmes considérés. Dans des développements futurs, d'autres classes de fonctions pourront être ajoutées facilement au système.

L'usager entre les fonctions sous forme d'expressions algébriques. Les opérandes de l'expression peuvent être des constantes ou des mots-clés qui indiquent les paramètres des fonctions. Tous les fonctions de la même classe ont le même jeu de paramètres (pour les détails, voir section III-3.3 du manuel de l'usager). A l'intérieur d'une classe, les fonctions sont numérotées de 1 à MFUNC. Une fonction est donc identifiée par l'identificateur de la classe suivi d'un numéro de fonction.

Voici la liste des opérateurs et fonctions intrinsèques qui peuvent être utilisés dans les expressions:

- jusqu'à 20 niveaux de parenthèses
- les opérateurs algébriques

+	addition
-	soustraction
*	multiplication
/	division
^	exponentiation
.MOD.	retenue de division
.MIN.	minimum
.MAX.	maximum

- les opérateurs relationnels (résultent en 0 si faux, 1 si vrai)

.EQ.	égal
.NE.	différent
.LT.	plus petit
.LE.	plus petit ou égal
.GT.	plus grand
.GE.	plus grand ou égal

- les opérateur logiques (un opérande zéro est considéré vrai, un opérande non-zéro est considéré faux)

.AND.	produit logique
.OR.	somme logique
.XOR.	ou exclusif

- les fonctions intrinseques

ABS()	valeur absolue
INT()	valeur entière tronquée
SQRT()	racine carrée
EXP()	fonction exponentielle
LN()	logarithme naturel
LGAM()	loggamma
SIN()	sinus
COS()	cosinus
ATAN()	arctangente

La précédence des opérateurs suit les conventions usuelles, sauf pour le fait que l'exponentiation a une associativité vers la gauche, i.e. A^B^C est évalué comme (A^B)^C, au lieu de la forme plus conventionnelle A^(B^C).

Prenons comme exemple encore une fonction volume-délai. L'usager pourrait définir une fonction BPR avec l'enoncé suivant:

```
FD1 = LENGTH * ( 2.5 + (VOLAU/(450*LANES))^3 )
```

où le mot-clef LENGTH représente la longueur du lien en km, VOLAU le volume auto sur le lien, et LANES le nombre de voies. Le résultat de la fonction est le temps de parcours du lien, en minutes.

De façon interne, les fonctions sont représentées comme une suite de paires opérande - opérateur. Un vecteur de nombres réels VFUNC contient les opérandes et un vecteur de nombres entiers IFUNC contient les codes

d'opérateur des fonctions d'une classe. De plus, le signe du code d'opérateur stocké dans IFUNC indique si l'opérande correspondant est une constante ou un paramètre. Dans le cas d'une constante, elle se trouve dans le mot correspondant de VFUNC, tandis que dans le cas d'un paramètre, VFUNC contient son adresse, i.e. un indice du vecteur des paramètres.

Cette méthode de représentation a l'avantage de servir directement à l'évaluation efficace des valeurs de fonctions et, en même temps, elle permet de reproduire exactement l'expression entrée par l'usager. Ceci ne serait pas le cas, si on utilisait, par exemple, une représentation interne de type post-fixe.

L'évaluation des fonctions est faite par un interprète simple, basé sur la précédence des opérateurs (Aho et Ullman, 1979). Dans ce but, on associe à chaque opérateur une *priorité gauche* et une *priorité droite*. Une opération est effectuée si la priorité gauche de l'opérateur correspondant excède ou est égale à la priorité droite de l'opérateur suivant, sinon opérande et opérateur sont empilés. Le tableau III-3-3 donne les priorités gauches et droites des opérateurs utilisés dans l'interprète de fonctions.

Opérateur(s):	Priorité gauche	Priorité droite
^	6	6
* / .MOD.	5	5
+ - .MIN. .MAX.	4	4
.EQ. .NE. .LT. .LE. .GT. .GE.	3	3
.AND. .OR. .XOR.	2	2
(f(	9	1
)	0	0

TABLEAU III-3-3: Les priorités gauches et droites des opérateurs.

Nos tests ont montré que la méthode d'évaluation de fonctions prend en moyenne de 2 à 3 fois le temps d'exécution d'une fonction FORTRAN qui évalue la même expression. Un tel facteur nous semble tout à fait acceptable, d'autant plus que, pour pouvoir utiliser une fonction compilée, il faut prendre la forme la plus générale de la fonction, tandis que dans notre approche chaque fonction peut être optimisée individuellement, par exemple en enlevant tous les termes égaux à zéro et les facteurs et exposants valant 1.

3.4 Les données auxiliaires

En plus des trois composantes principales des données (le réseau, les matrices et les fonctions), la banque de données contient certaines données auxiliaires, qui sont importantes pour le fonctionnement de EMME/2, même si elles ne sont pas reliées directement avec une des trois composantes principales.

Un *journal de bord* maintient une trace chronologique des derniers 200 à 300 appels de modules. Pour chaque module appelé, on y trouve le numéro de module, le scénario courant, les initiales de l'usager, la date, ainsi qu'un commentaire. Pour la plupart des modules, les commentaires sont optionnels; ils peuvent être entrés lors de l'appel du module. Pour certains modules-clefs, comme les manipulations de scénarios et les affectations, des messages standard sont générés, donnant les détails de l'opération effectuée. Le journal de bord permet donc, au moins en partie, de reconstruire l'histoire récente d'une banque de données. Puisque le journal de bord est réalisé par un tampon cyclique de taille fixe, les entrées les plus anciennes sont éliminées pour faire place aux entrées actuelles. Le nombre d'entrées dépend de la longueur des commentaires.

Les *démarcations* sont des références géographiques qui peuvent être superposées aux dessins de réseau, afin de les rendre plus facilement lisibles. L'usager peut définir jusqu'à 9 ensembles de démarcations, numérotés de 1 à 9. Toutes les démarcations du même ensemble sont tracées en même temps. Une démarcation est représentée par une suite de points sur le plan du réseau. Selon le type de démarcation, ces points sont dessinés d'une façon isolée, comme des lignes interpolées linéairement ou comme des

courbes interpolées par une fonction spline. Pour que l'usager ne puisse pas confondre les démarcations avec le contenu proprement dit du dessin, elles sont représentées graphiquement par des symboles (· - = + x > Δ) de taille et intervalle variables. Le contenu logique des démarcations est entièrement la responsabilité de l'usager. Voici quelques exemples typiques d'utilisation des démarcations:

- barrières géographiques (rivières, lacs, bord de la mer)
- frontières politiques (communauté, province, état)
- délimitation des zones origines / destinations
- emplacement de certains services spéciaux (points de comptage, points de péage, hôpitaux, pompiers, police)

Les autres données auxiliaires comprennent la *table des équipements* (voir section 2.3), les *paramètres de module* (voir section 2.5) et les *fenêtres prédefinies par l'usager* (voir section 4.4).

4. Aspects graphiques

4.1 Le choix d'un système graphique de base

Les graphiques qui sont générés par le logiciel EMME/2 sont principalement des dessins de réseaux. Les éléments de base d'un tel dessin sont les noeuds et les liens du réseau, possiblement annotés par des attributs graphiques ou numériques. Le dessin typique est donc très simple pour ce qui est de la complexité de la représentation graphique. On n'a pas besoin de représentations tri-dimensionnelles, d'algorithmes de lignes cachées, ni de remplissage de surfaces compliquées. Par contre, en termes de taille et quantité d'information, la complexité d'un dessin est généralement très élevée. Le réseau d'une ville moyenne peut comporter de 500 à 2500 noeuds et 1500 à 8000 liens; il est tout-à-fait raisonnable de vouloir afficher tout le réseau sur le même dessin, en indiquant pour chaque lien la valeur d'un attribut en format graphique ou numérique.

Bien que la plupart des graphiques dans EMME/2 soient des dessins de réseau, il y a aussi d'autres types de graphiques, comme des histogrammes, des graphes X-Y et des détails de carrefours aux intersections.

Comment satisfaire ces besoins d'affichage graphique? Une décision importante, qu'il a fallu prendre au tout début du développement du système EMME/2. Est-ce qu'on utilise un système graphique existant? Si oui, lequel? Ou est-ce qu'on se lance à développer notre propre système graphique de base?

Une multitude de critères influençaient ce choix. Voici les besoins les plus importants qui ont été identifiés:

a) Le langage de programmation du système graphique choisi doit être compatible avec le langage utilisé pour le développement de EMME/2.

b) Le système graphique doit permettre l'utilisation de terminaux graphiques et tables traçantes de différents manufacturiers.

c) Pour pouvoir garder la portabilité du système EMME/2, il est nécessaire que le système graphique aussi soit transférable à une multitude d'ordinateurs différents. En particulier, cela devrait inclure des micro-ordinateurs de haut de gamme.

d) Le jeu de primitives offert par le système graphique doit être adéquat. Un système graphique trop limité freinera le développement de EMME/2; un système trop complexe et surchargé, avec des options inutiles dans notre contexte, ne serait qu'un alourdissement.

e) EMME/2 est un système interactif-graphique. L'interactivité implique que l'usager qui demande un dessin, s'attend (et a le droit de s'attendre) à une réponse immédiate. L'efficacité du système graphique prend donc une importance cruciale, ceci surtout en vue de la grande quantité d'informations contenue dans un dessin.

A ces besoins généraux s'ajoutent les contraintes imposées par la disponibilité (ou non-disponibilité) de certaines ressources. La contrainte la plus sévère étant l'espace mémoire disponible, en mode d'opération interactif, sur l'ordinateur CDC CYBER 173 de l'université de

Montréal: programme et données ne peuvent dépasser 32768 mots de mémoire - une limite très restreignante qui a influencé beaucoup d'aspects du développement de EMME/2.

A la lumière de ces besoins et contraintes, comment se comparaient les divers systèmes graphiques disponibles? Voici la liste des principaux candidats qui ont été considérés pour le système graphique de base de EMME/2:

MIRA: Un langage graphique basé sur le langage de programmation PASCAL, développé à l'Université de Montréal (Magnenat-Thalmann et Thalmann, 1981).

PLOT10: Un ensemble de sous-routines écrites en FORTRAN qui permettent l'utilisation des équipements graphiques TEKTRONIX. Ce logiciel est distribué par la compagnie TEKTRONIX.

IGR: Un ensemble de sous-routines écrites en FORTRAN qui permettent l'utilisation d'équipements graphiques quelconques. Ce logiciel a été développé à l'EPFL à Lausanne (Crvcanin, 1978).

Tous ces logiciels sont des systèmes graphiques à but général qui essayent de satisfaire les besoins typiques de tout type d'application graphique. A la rigueur, bien sûr, on peut se servir de n'importe quel de ces systèmes pour produire des dessins de réseaux. Par contre, tous ces système montrent de sérieux défauts quant à leur capacité de répondre aux besoins mentionnés ci-haut.

L'utilisation du langage graphique MIRA était exclue principalement à cause du besoin a). On avait des trè bonnes raisons de faire le

développement de EMME/2 en FORTRAN. La disponibilité de MIRA en soi n'était pas une raison suffisante pour justifier l'utilisation de PASCAL au lieu de FORTRAN. Le point fort de MIRA, la création et la manipulation d'objets graphiques de plus haut niveau, ne s'applique guère aux dessins de réseaux de transports. On peut définir le réseau entier comme un objet, ou on définit chaque élement de base (noeud, lien) comme un objet. Dans le premier cas on n'a qu'un seul objet, beaucoup trop grand pour être traité d'une façon intégrale; dans le deuxième cas, le nombre d'objets dépasse de loin la capacité du système.

Le logiciel PLOT10 est parmi les plus utilisés au monde. Il s'agit d'un système graphique traditionnel, i.e. un ensemble de sous-routines qui permettent d'appeler les différentes primitives graphiques supportées par le terminal graphique. Son désavantage majeur est le fait qu'il ne fonctionne qu'avec les produits de la compagnie TEKTRONIX - point faible quant au besoin b) ci-haut.

Finalement, le système IGR, composé aussi d'un ensemble de routines graphiques écrites en FORTRAN, a l'avantage de pouvoir supporter, au moins en principe, toutes sortes d'équipements graphiques. Sa disponibilité par contre, quant aux différents types d'ordinateurs, est restreinte aux ordinateurs CYBER et VAX, à cause de très fortes dépendances sur le système d'exploitation. Quant à l'environnement informatique par contre, le logiciel doit être adapté à chaque installation, parce qu'il dépend fortement du système d'exploitation. Présentement, il n'existent que des versions pour CDC Cyber et VAX (sous VMS).

Après avoir évalué les systèmes graphiques existants dont nous avions

connaissance, il était clair qu'aucun des systèmes ne constituait la solution idéale à nos besoins. Bien entendu, la raison n'était pas que ces systèmes ne permettaient pas de générer des dessins de réseaux de style EMME/2. Au contraire, l'utilisation de ces systèmes était rejetée parce qu'ils étaient considérés trop généraux pour nos besoins restreints, offrant beaucoup d'options dont on savait qu'elles ne seraient jamais utilisées dans le contexte de EMME/2. A cause de leur généralité, ces systèmes utilisent considérablement plus d'espace mémoire et exécutent les tâches simples d'une façon moins efficace.

La conclusion: écrire nous-mêmes un système graphique de base qui répondrait exactement à nos besoins. Cette approche avait en plus l'avantage de nous permettre de définir les primitives telles qu'elles favorisent le type de dessins le plus pertinent dans notre contexte: le dessin d'un réseau.

C'est comme cela que, dans la première phase du développement de EMME/2, un ensemble de routines graphiques, appelé QPLOT, fut écrit. Il s'agit d'un système graphique minimal, qui est optimisé pour l'affichage de dessins de réseaux. Il ne constitue que la couche de plus bas niveau des routines et modules graphiques de EMME/2. Les routines QPLOT sont présentées avec plus de détails dans la section suivante.

4.2 QPLOT - le logiciel graphique de base

QPLOT est un ensemble de routines graphiques de bas niveau écrit en FORTRAN. Bien que ces routines peuvent servir pour générer n'importe quel genre de dessins, ses primitives graphiques permettent un traitement particulièrement efficace des dessins de réseaux.

Qu'est-ce qui distingue le dessin d'un réseau de transport d'autres types de dessins, comme par exemple des dessins d'objets physiques?

La différence principale se trouve dans la notion d'échelle. Un dessin d'un objet physique ne connaît qu'une seule échelle. Un agrandissement d'une partie du dessin est accompli en agrandissant chaque élément du dessin par le *même* facteur d'échelle.

Ceci n'est pas le cas pour un dessin de réseau. Ici il faut distinguer deux échelles différentes: L'échelle géographique et l'échelle des attributs. *L'échelle géographique* détermine l'emplacement des objets (i.e. des noeuds). *L'échelle des attributs* détermine la taille des symboles dessinés (i.e. boites de noeuds, largeurs des bandes qui représentent les liens).

Cette distinction devient plus claire en regardant l'exemple d'une opération graphique, appelée "zoom-in", le réaffichage agrandi d'une partie du dessin déjà sur l'écran. La figure III-4-1 montre le dessin original qui représente les volumes auto prédits pour un scénario du réseau de Winnipeg. Le dessin est très lisible dans les parties moins denses, mais les détails de la partie centrale, représentant le centre-ville, se perdent

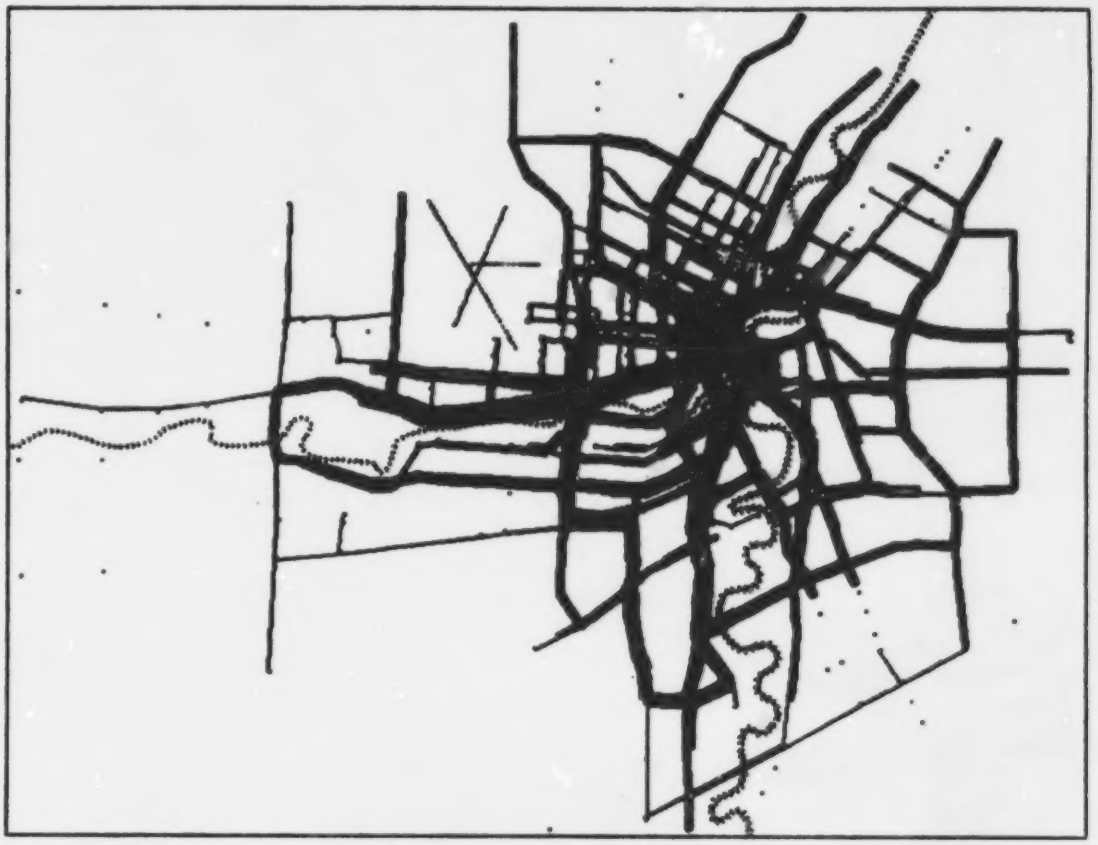

FIGURE III-4-1: Représentation proportionnelle des volumes-autos.

dans une grande tache noire. Pour pouvoir discerner les volumes auto au centre-ville, il faudrait agrandir la partie centrale du dessin. La figure III-4-2 montre un agrandissement "ordinaire" de cette partie, i.e., l'agrandissement a été produit en applicant le même facteur d'échelle à tous les éléments affichés. La figure III-4-2 montre bien que ce type d'agrandissement ne fait pas ce que nous désirons. Au lieu d'afficher les détails de ce qui apparaissait comme une tache illisible dans le dessin original, cet agrandissement produit encore la même tache noire, aussi illisible que dans l'original, mais en plus grand. C'est ce mécanisme de "zoom" qui est disponible dans tous les systèmes graphiques commerciaux, et aussi comme opération locale sur les terminaux graphiques évolués. Comme on le voit, son utilité dans le contexte d'affichage de réseaux n'est que

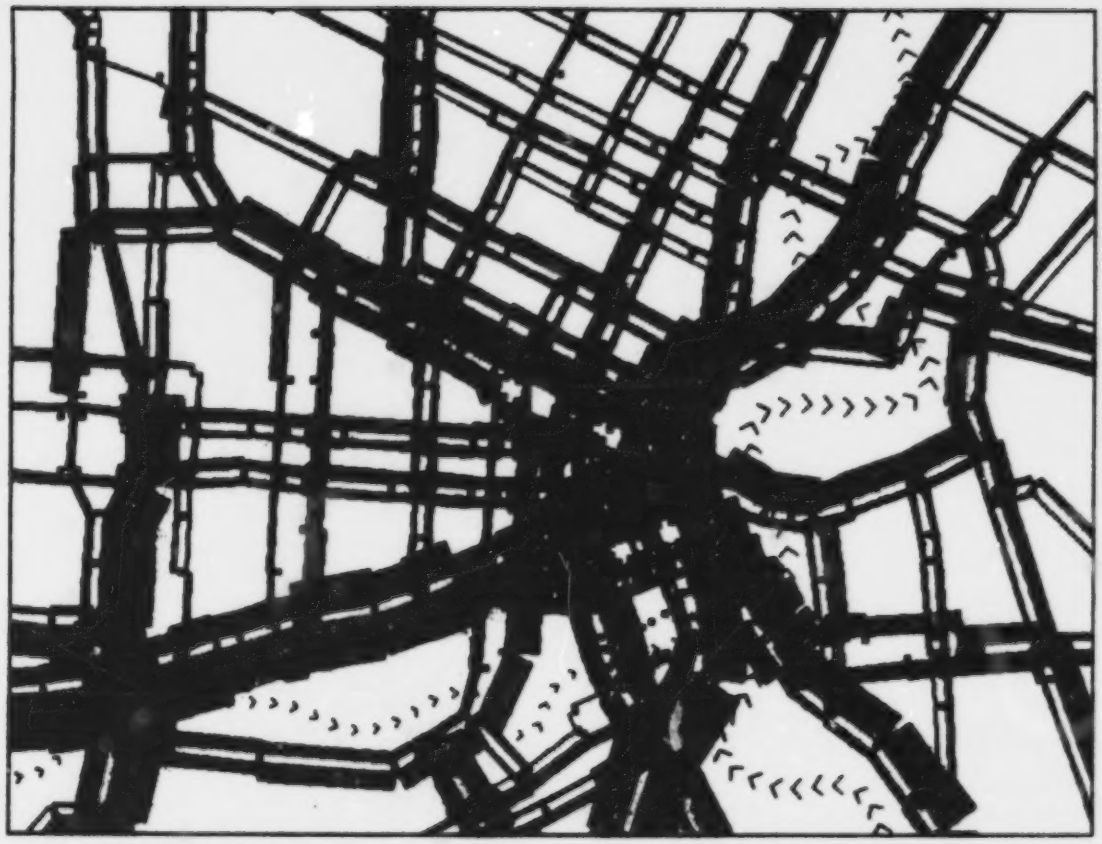

FIGURE III-4-2: Agrandissement "ordinaire" du centre-ville.

très limitée. Pour être utile, le mécanisme de "zoom" doit changer l'échelle géographique indépendemment de l'échelle des attributs. Ceci est montré dans la figure III-4-3: seulement l'échelle géographique est changée, et l'échelle des attributs est restée la même. Les détails des volumes auto du centre-ville sont maintenant devenus très lisibles.

Voici donc le principe de base utilisé dans les routines QPLOT:

- L'emplacement d'un objet est déterminé par sa position dans un système de *coordonnées de l'usager*.

FIGURE III-4-3: Agrandissement par changement de l'échelle géographique.
- La taille des objets dessinés est donnée en unités de *coordonnées normalisées de l'écran.*

Le *système des coordonnées de l'usager* est utilisé pour donner la position de l'objet dans l'espace réel, et il est, comme son nom l'indique, donné par l'usager. En général, il s'agit du même système de coordonnées qui a été utilisé pour géocoder le réseau. L'unité de coordonnée correspond habituellement à l'unité de distance, par défaut le kilomètre. La partie visible du plan de coordonnées de l'usager est appelée la *fenêtre* (en anglais "window"). Il s'agit d'une région rectangulaire spécifiée par les coordonnées des deux coins en bas à gauche et en haut à droite.

Le système de coordonnées normalisées de l'écran est utilisé pour déterminer l'emplacement d'un objet dans l'espace du dessin (écran ou feuille de papier). L'axe des X s'étend horizontalement de gauche à droite et l'axe des Y s'étend verticalement du bas vers le haut. Pour une application donnée, on suppose que les dimensions de l'écran, en coordonnées normalisées, restent fixes. Les dimensions physiques peuvent changer et sont dépendantes de l'équipement graphique utilisé. A chaque instant, seulement une partie rectangulaire de l'écran, appelé le *cadre*, (en anglais "view port") est active, i.e. peut être modifiée par des opérations graphiques. Le cadre correspond à l'image de la fenêtre définie sur le plan des coordonnées de l'usager. Une condition importante est satisfaite si la fenêtre et le cadre ont le même *ratio d'aspect*, i.e. quotient de la hauteur divisé par la largeur. Seulement dans ce cas la transformation de coordonnées de l'usager aux coordonnées de l'écran est *conforme*, c'est-à-dire conserve les angles. Evidemment cette condition est aussi importante pour l'affichage d'un réseau que pour la plupart des autres applications graphiques.

Pour EMME/2 on a choisi un écran-type en format "paysage", avec les dimensions normalisées

$$0 \leq X < 1024,$$

$$0 \leq Y < 768.$$

Le ratio d'aspect est donc de 4X:3Y, le format trouvé dans la plupart des terminaux graphiques contemporains. Il est à noter que les coordonnées normalisées admettent des valeurs réelles. Le choix des dimensions ci-haut ne limite donc pas la précision du dessin. Seulement à la toute dernière étape ces coordonnées sont elles transformées et tronquées afin de tenir compte de la résolution spécifique de l'équipement utilisé.

Ce qui suit est un résumé des routines principales du système graphique de base QPLOT.

QPLOT: La routine principale de l'ensemble. Elle sert à plusieurs fonctions et contient, comme variables statiques locales, l'information nécessaire pour décrire l'état du système. Ceci comprend la définition de la fenêtre (en coordonnées de l'usager), la définition du cadre (en coordonnées normalisées), ainsi que les coefficients des fonctions de transformation dans les deux sens. La tâche principale de cette routine est de tracer un vecteur qui relie deux points. Les points sont spécifiés en coordonnées de l'usager. QPLOT s'occupe de la transformation des points en coordonnées normalisées, du découpage du vecteur en parties visibles et invisibles ("clipping"), et, si la partie visible est non-vide, de son affichage en utilisant un patron de ligne choisi. Un indicateur est retourné, qui indique si le vecteur se situe complètement, partiellement ou pas du tout dans la fenêtre actuelle. Selon le mode d'appel, une des opérations auxiliaires suivantes peut aussi être demandée:

a) Définition du cadre d'une façon absolue. Les coordonnées du cadre sont spécifiées en coordonnées normalisées.

b) Définition de la fenêtre d'une façon absolue. Les coordonnées de la fenêtre sont spécifiées en coordonnées de l'usager.

c) Définition du cadre avec ajustement à la fenêtre (qui est déjà définie). Si le cadre donné n'a pas le même ratio d'aspect que la fenêtre, un cadre diminué avec le même ratio est calculé.

d) Définition de la fenêtre avec ajustement au cadre existant. Si le ratio d'aspect de la fenêtre ne coïncide pas avec celui du cadre, une fenêtre agrandie avec le même ratio est calculée.

e) Le cadre actuel est retourné au programme appelant.

f) La fenêtre actuelle est retournée au programme appelant.

g) Transformation d'un point de coordonnées de l'usager en coordonnées normalisées.

h) Transformation d'un point de coordonnées normalisées en coordonnées de l'usager.

i) Découpage de la partie visible d'un vecteur. La partie visible (celle contenue dans la fenêtre) d'un vecteur, qui est décrit par ses deux points terminaux (en coordonnées de l'usager), est retournée en coordonnées de l'usager.

QRIGHT: Affichage d'un vecteur décalé à droite par rapport aux deux points spécifiés. Les points sont donnés en coordonnées de l'usager. Le décalage est spécifié en unités de coordonnées normalisées. Un nombre négatif indique un décalage à gauche.

QBAR: Affichage d'une bande à largeur variable reliant deux points, qui sont spécifiés en coordonnées de l'usager. La largeur de la bande et son décalage à droite sont spécifiés en unités normalisées.

QBOX: Affichage d'une boite rectangulaire centrée à un point donnée en coordonnées de l'usager. La largeur et la hauteur de la boîte sont spécifiées en unités normalisées.

QCIRC: Affichage d'un cercle ou d'un polygone régulier. Un polygone régulier à N cotés est dessiné. Le centre est spécifié en coordonnées de l'usager et le rayon est spécifié en unités normalisées. Si la valeur N n'est pas spécifiée, une valeur par défaut est calculée automatiquement en fonction du rayon, d'une manière qui assure que le polygone soit perçu comme un cercle.

QARC: Un arc de cercle est affiché. Le centre ainsi que le point de début de l'arc sont spécifiés en coordonnées de l'usager et l'angle est donné en radians. Un angle positif indique une rotation dans le sens horaire, un angle négatif indique une rotation dans le sens anti-horaire, autour du centre.

QSPLIN: Affichage d'une courbe "spline" qui relie deux points donnés. A chaque point, un vecteur de direction est donné qui indique la dérivée desirée de la courbe à ces points. Ceci est une forme particulière d'une courbe "spline", qui est discutée en détail dans la section 6 de ce chapitre.

QTEXT: Affichage d'une chaîne de caractères. La position du début du texte et la direction d'écriture sont spécifiées en coordonnées de l'usager, tandis que les attributs des caractères (hauteur, largeur, espacement et inclinaison) sont spécifiés en unités normalisées. Le générateur de caractères qui est utilisé pour les équipements qui ne peuvent pas générer des caractères dans n'importe quelle direction et taille est discuté en détail dans la section 6.

En plus de ces routines graphiques principales, QPLOT comprend aussi

un ensemble de routines d'interface ("driver"), qui s'occupent de l'interface entre les routines décrites ci-haut et l'équipement utilisé. Ces routines sont spécifiques à chaque classe d'équipement utilisé. Présentement deux versions existent: la première couvre toute la gamme des équipements graphiques compatibles avec les produits TEKTRONIX série 4000, et la deuxième fait l'interface avec le terminal graphique couleurs CROMATICS modèle CGC 7900. Ce qui suit est une description brève des routines d'interface.

TKINIT: Cette routine initialise l'équipement pour un nouveau dessin. Pour un terminal graphique, cela implique l'effaçage de l'écran et l'initialisation des paramètres pertinents.

TKPLOT: Cette routine s'occupe du traçage proprement dit. A partir de la position courante, un trait, visible ou non-visible, est tracé jusqu'à un autre point de l'écran. Toutes les coordonnées sont spécifiées en unités normalisées. Les traits visibles peuvent être tracés avec un de plusieurs patrons de ligne distincts. Le nombre et le type de ces patrons dépendent de l'équipement utilisé.

TKCROS: Ceci est la routine de base pour accomplir l'entrée graphique. Si l'équipement est muni d'un dispositif pour l'entrée d'une position graphique, cette routine s'occupe d'activer ce mécanisme. Les coordonnées du point désigné par l'usager sont retournées, ainsi que le caractère tapé par l'usager pour indiquer la fin de l'opération. Selon le type d'équipement, le dispositif d'entrée graphique peut être des potentiomètres X-Y (en anglais "thumb wheels"), un pivot X-Y ("joy-stick"), une souris ("mouse"), un crayon-pointeur ("light pen") ou une tablette graphique. Puisque

cette routine est le seul moyen d'entrée graphique dans QPLOT, elle est utilisée pour placer des objets ("locator"), sélectionner des objets ("pick") et comme moyen analogique de spécifier une valeur numérique ("valuator").

TKTEXT: Cette routine place une chaîne de caractères à une position donnée sur l'écran. Seulement l'écriture horizontale est admise et la dimension des caractères est limitée à un des quatre formats standards. Si l'équipement ne peut pas générer lui-même ce format de caractères, le générateur de caractères interne est utilisé (voir section 6 de ce chapitre).

TKPOS: Le curseur est positionné à l'endroit spécifié et l'équipement est mis en mode alphanumérique, prêt à accepter une chaîne de caractères entrée par l'usager.

TKCOL: La couleur d'affichage est changée pour la couleur indiquée. Cette couleur s'applique en même temps aux vecteurs et aux chaînes de caractères. Le nombre de couleurs est considéré infini. Dans une implantation réelle, cela signifie qu'a partir d'un indice de couleurs qui dépend de l'équipement, les couleurs se répètent. Voir plus loin dans cette section pour une description de la façon dont les couleurs sont utilisées dans EMME/2.

En rétrospective, il apparaît que la décision de développer QPLOT, au lieu d'utiliser un système graphique général existant, a montré plusieurs avantages:

- Le développement de QPLOT n'a pas pris plus de temps ou de ressources qu'une adaptation à un système existant.

- QPLOT s'est révélé être un outil de dessin très efficace. Pour les tâches auxquelles il était destiné, il se compare très favorablement avec les autres systèmes, en terme d'efficacité.

- Le système de base de QPLOT n'occupe qu'un très modeste espace de mémoire centrale. Son implantation sur CYBER totalise environ 5k mots (dont une grosse partie pour le générateur interne de caractères).

- Le transfert de QPLOT à d'autres types d'ordinateurs s'est avéré très simple, puisque toutes les routines sont écrites entièrement en FORTRAN.

4.3 Le dessin

Le système EMME/2 comprend une vingtaine de modules graphiques, qui génèrent un total de plus de cinquante différents types de sorties graphiques. On perdrait vite tous les avantages d'un système interactif-graphique, si on demandait à l'usager de mémoriser cinquante différentes façons de générer, d'interpréter et de manipuler des sorties graphiques. Il va de soi qu'une standardisation est une nécessité absolue pour arriver à une interface homme - machine efficace. Il faut que l'usager n'ait qu'à apprendre un petit ensemble de règles simples, qui s'appliquent uniformément à tous les modules. Ces règles doivent comprendre la génération d'un dessin, son interprétation et toutes les manipulations graphiques subséquentes. Le processus d'apprentissage devient simple et l'usager se sent à l'aise avec le système entier, seulement si tous les modules se comportent d'un façon consistante.

Il est essentiel que ces règles soient implantées strictement. Même

si, en soi, elle peut être très utile pour un module particulier, une option non-standard peut résulter en frustration pour l'usager, soit, parce qu'il ne se souvient pas de cette exception, soit, parce qu'il prend pour acquis qu'il peut l'appeler dans n'importe quel module. Les options prises en considération n'étaient donc que celles qui pouvaient être réalisées dans tous les modules pertinents, à travers tout le système. En conséquence, certaines options n'ont pas été implantées, puisque, pour des raisons techniques, elles n'auraient pu être réalisées que dans des modules particuliers.

La standardisation des modules graphiques a d'autres avantages qu'il ne faut pas oublier: elle facilite énormément la conception, la programmation et la documentation du système.

Dans EMME/2 on distingue deux formats de sortie graphique, *le dessin* (en anglais, "plot") et *la feuille de travail* ("graphic work sheet"). Ces deux formats sont nécessaires pour tenir compte des deux modes d'utilisation d'un système graphique: le format dessin est utilisé pour toute sortie graphique qui a comme but d'afficher et de permettre d'interroger des éléments contenus dans la banque de données, tandis qu'on se sert de la feuille de travail pour toute modification interactive-graphique du contenu de la banque de données. Dans ce qui suit, une brève description des deux formats graphiques est donnée en expliquant les règles associées avec chacun.

La figure III-4-4 est un exemple d'une sortie graphique en format *dessin*. L'écran est divisé en six parties, dénotées A à F dans la figure III-4-4. Chacune de ces sous-divisions contient un aspect particulier de l'information nécessaire pour pouvoir interpréter correctement le dessin.

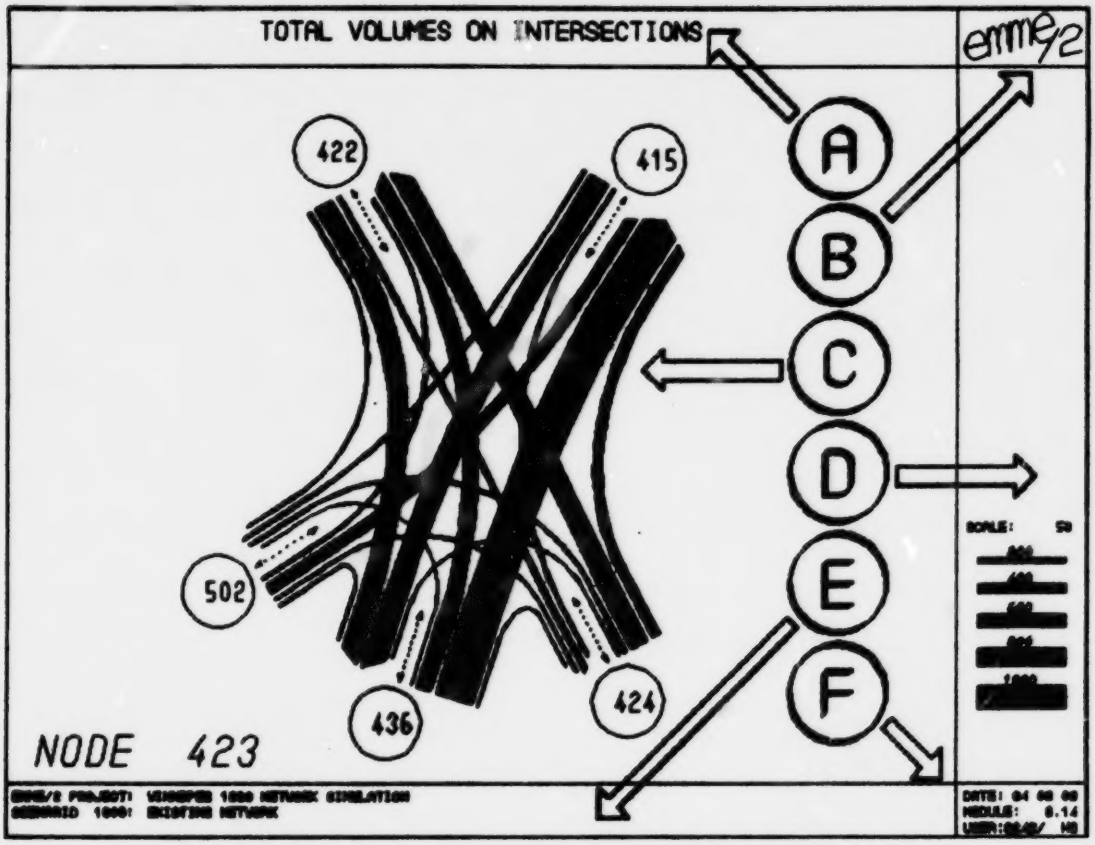

FIGURE III-4-4: Format général d'un dessin.

Voici le rôle associé avec chacune des sous-divisions:

A) En haut de l'écran (ou de la page, dans le cas d'un dessin produit sur papier), on trouve toujours une description du *type de dessin*. Elle indique le type d'attributs et d'éléments affichés. Cette information est générée par le module même et ne peut être modifiée par l'usager.

B) Dans le coin supérieur droit apparaît le *logo* de EMME/2.

C) La *partie principale* du dessin contient la sortie graphique proprement dite. Le ratio d'aspect de 4X:3Y est maintenu; les dimensions de cette partie sont de 900 par 675 unités normalisées, soit environ 78% de la surface disponible. Si l'information contenue dans cette partie est

affichée sous forme de réseau, le dessin est appelé un *dessin de réseau*. Certaines options, discutées plus bas, ne s'appliquent que pour les dessins de réseau.

D) La marge de droite de l'écran est appelée la *légende*. Elle contient toute l'information nécessaire à l'interprétation des détails du dessin. On trouve ici des informations sur le sous-ensemble d'éléments selectionnés, l'échelle des attributs (si le dessin contient des attributs représentés en forme graphique) et, pour tout dessin de réseau, les coordonnées qui définissent la fenêtre affichée.

E) En bas de l'écran, se trouvent des informations portant sur la *provenance* des données affichées. Une à trois lignes contiennent une description du projet (qui permet d'identifier la banque de données), et les descriptions du(des) scénario(s) et / ou de la (des) matrice(s) pertinente(s). L'espace libre à droite de ces descriptions est utilisé pour l'entrée des commandes graphiques, comme décrit plus loin dans cette section.

F) Finalement, le coin inférieur droit identifie la *date*, le numéro du *module* qui a généré le dessin, l'identificateur de l'*installation* et les initiales de l'*usager*.

Il est à noter que cette composition de la page est tout-à-fait arbitraire. Beaucoup d'autres arrangements sont possibles et le choix pris pour EMME/2 a été finalement une question de goût. L'importance ici se trouve plus dans le fait d'avoir établi un tel arrangement fixe, dès le début du développement. En fait, l'arrangement décrit ci-haut fut conçu, sur une feuille de papier, avant même que le système graphique QPLOT soit

opérationnel. Il n'a subi que des changements très mineurs depuis, et a contribué beaucoup à la "personnalité" uniforme du logiciel EIME/2.

Quelle est la séquence des événements permettant de générer un tel dessin?

Premièrement, l'usager doit répondre à une suite de questions qui lui sont posées en mode de dialogue. Ces questions permettent d'établir l'ensemble des données à afficher, ainsi que les paramètres graphiques. S'il s'agit d'un dessin de réseau, une fenêtre automatique est calculée, qui contient par défaut tout le sous-réseau spécifié. Si l'usager désire utiliser une fenêtre différente, il peut la spécifier par une des méthodes suivantes:

- la fenêtre est donnée par les coordonnées des coins inférieur gauche et supérieur droit;
- la fenêtre est calculée en fonction d'un ensemble de noeuds qui doivent y être inclus;
- la fenêtre est centrée sur un noeud avec un diamètre donné;
- la fenêtre correspond à une des 26 fenêtres, dénotées de "a" à "z", qui ont été prédéfinies par l'usager;
- on utilise la même fenêtre que celle utilisée pour le dernier dessin de réseau.

Pour les dessins qui contiennent la représentation graphique d'un attribut de façon proportionnelle, comme par exemple un dessin de volumes, l'usager doit spécifier un facteur d'échelle pour l'attribut. Celui-ci est

donné en unités de l'attribut par unité de coordonnée normalisée.

Après que l'usager ait parcouru le dialogue, le contenu du dessin est complètement déterminé. Il ne reste qu'à l'afficher. La sortie peut être acheminée vers deux destinations: directement à l'écran d'un terminal graphique, ou bien vers le *fichier des dessins*, où les dessins sont emmagasinés pour utilisation ultérieure (table traçante, imprimante graphique ou reproduction photographique). Dans ce qui suit, nous allons supposer que le dessin est affiché directement à l'écran d'un terminal graphique, ce qui est de loin le cas le plus fréquent.

L'affichage d'un dessin peut prendre de quelques secondes à plusieurs minutes, dépendant de la complexité du dessin et de la vitesse de transmission entre l'ordinateur et le terminal graphique. Pendant que le dessin est en train de s'afficher, l'usager peut, s'il le juge nécessaire, envoyer une interruption au système. Le dessin sera alors interrompu à la prochaine occasion, habituellement avant le début du traitement du prochain élément à afficher (ex. prochain noeud ou prochaine ligne de transport en commun). Pour confirmer que le dessin a été interrompu, et est donc à considérer incomplet, un astérisque (*) est placé au coin supérieur gauche de l'écran. L'usager peut maintenant continuer de la même façon, comme si le dessin avait été complété normalement. Il est à noter que ce mécanisme d'interruption n'est pas disponible sur toutes les installations, puisqu'il requiert l'accès à une routine de détection d'interruption, ce qui n'est pas possible sur tous les systèmes d'exploitation.

Une fois le dessin complété, l'usager est averti par le son de la cloche. Le curseur alphanumérique apparaît à la position des commandes graphiques, qui se trouve dans la partie droite de la sous-division e)

décrite ci-haut. Le module est maintenant prêt à accepter des *commandes graphiques*.

4.4 Les commandes graphiques

Les *commandes graphiques* servent à modifier et manipuler le dessin, une fois qu'il est affiché à l'écran. Puisqu'à ce moment-là, l'écran contient un dessin fini, il faut garder la procédure des commandes graphiques aussi peu bavarde que possible, sinon on risque de gâter le contenu de l'écran. C'est pour cette raison, qu'une commande graphique est invoquée par l'entrée d'un seul caractère. Si l'équipement le permet, ce caractère n'apparaît même pas à l'écran.

Certaines commandes, comme par exemple un changement de la fenêtre, causent le réaffichage automatique du dessin modifié par la commande. On dit que ces commandes causent un "break". Plusieurs commandes graphiques peuvent être entrées sur la même ligne; dans ce cas, elles sont exécutées, l'une après l'autre, dans la séquence indiquée. Si, sur la même ligne, il y a plusieurs commandes qui causent un "break", le dessin n'est affiché qu'une seule fois, après l'exécution de toutes les commandes.

Pour la même raison (i.e. ne pas endommager le dessin existant), la détection d'une erreur pendant l'appel ou l'exécution d'une commande graphique ne provoque pas un message d'erreur, comme ailleurs dans le système. Trois sons de cloche avertissent l'usager que la commande est invalide et a été ignorée. Un usager qui a besoin d'information détaillée sur la nature exacte de l'erreur doit consulter le fichier des erreurs (voir aussi section 2 de ce chapitre).

Voici une liste des commandes graphiques et de leurs fonctions. La liste est en ordre alphabétique des caractères qui représentent la commande. On note que ces caractères ont été choisis comme abréviation des termes correspondants anglais, indiqués entre parenthèses. Une description plus détaillée des commandes graphiques se trouve dans la section III.6.2 du manuel de l'usager de EMME/2.

C: ("Continue on same plot") Cette commande est utilisée dans les modules qui affichent un dessin en étapes, en donnant à l'usager la possibilité d'intervention entre les étapes. Des exemples se trouvent dans les modules 2.15 (affichage interactif de chemins plus courts) et 6.15 (analyse d'accessibilité).

M: ("Manual mode") On entre dans un mode de modifications manuelles du dessin. Des ajouts manuels, en forme de texte ou de suite de vecteurs, peuvent être superposés au dessin affiché. Il est à noter que cette commande n'est qu'une aide pour annoter manuellement un dessin, elle n'entraîne aucune modification du contenu de la banque de données.

N: ("Node Number toggle") Cette commande alterne entre l'affichage des noeuds avec ou sans leur numéro. Elle cause un "break" et, évidemment, ne s'applique qu'aux dessins de réseau.

P: ("Send to Plot file") Le dessin courant est regénéré et redirigé vers le fichier des dessins pour utilisation ultérieure.

Q: ("Quit") On termine l'exécution du module courant pour retourner au menu principal.

R: ("Restart") On termine cette séquence de dessins et on recommence au début du module courant.

S: ("Change Scale") L'usager demande un changement de l'échelle des attributs. Il doit, en conséquence, spécifier un facteur de changement; celui-ci affecte, d'une façon proportionnelle, la taille des attributs affichés. Cette commande cause un "break". Elle ne s'applique qu'aux dessins qui contiennent la représentation proportionnelle d'un attribut.

W: ("Change network Window") Cette commande permet à l'usager d'effectuer un changement de la fenêtre, i.e. de la région du réseau qui est affichée à l'écran. Des sous-commandes, expliquées en détail dans le manuel de l'usager de EMME/2, permettent la spécification de la nouvelle fenêtre selon une des méthodes suivantes:

- La nouvelle fenêtre est spécifiée à l'aide du curseur graphique ("zoom in"). L'usager pointe sur les coins inférieur gauche et supérieur droit de la fenêtre désirée.

- A l'aide du curseur graphique, l'usager indique le point sur le réseau qui constitue le centre de la nouvelle fenêtre ("pan"). Les dimensions de la fenêtre restent inchangées;

- La nouvelle fenêtre est adjacente à la fenêtre actuelle. Cette option permet de se "promener" dans le réseau en affichant la fenêtre voisine à droite, à gauche, en haut ou en bas.

- La nouvelle fenêtre est obtenue en agrandissant la fenêtre actuelle par un certain facteur ("zoom out").

- L'usager dispose d'un jeu de 26 fenêtres prédéfinies, dénotées de "a" à "z". Ces fenêtres peuvent être créées et rappelées à l'intérieur de la commande graphique "W".

- Le système garde une pile des dix fenêtres qui ont été utilisées le plus récemment. Une sous-commande spéciale permet de remonter à la dernière fenêtre.

La commande "W" cause un "break" et elle ne s'applique qu'aux dessins de réseau.

1-9: L'ensemble des lignes de démarcation N (1 à 9) est superposé au dessin. Cette commande ne s'applique qu'aux dessins de réseau.

4.5 La feuille de travail graphique

Portons maintenant notre attention au deuxième format d'affichage graphique, à la *feuille de travail graphique*. Ce format est utilisé dans tous les modules qui permettent la *modification interactive-graphique* des éléments contenus dans la banque de données. Contrairement au dessin, qui sert principalement à la sortie graphique, l'*entrée graphique* joue pour la feuille de travail un rôle aussi important que la *sortie graphique*. C'est en combinant les deux qu'on obtient un outil flexible pour l'édition interactive-graphique.

Le nom "feuille de travail" a été choisi pour souligner que ce type d'affichage ne produit pas de "beaux dessins" qui peuvent être mis directement dans un rapport technique. Il s'agit plutôt d'une page brouillon. L'usager s'en sert en ajoutant de nouveaux éléments et / ou en rayant les

élément dont il n'a plus besoin. Pendant une session de travail, cette feuille accumule donc l'histoire des modifications effectuées. Si, après plusieurs modifications, la feuille de travail commence à devenir moins lisible, c'est à l'usager de demander une nouvelle feuille de travail. L'écran est alors effacé et une feuille de travail vierge est affichée qui tient déjà compte de toutes les modifications déjà faites.

La feuille de travail reconnaît les quatre opérations de base suivantes:

- ajout d'un nouvel élément;
- élimination d'un élément existant;
- modification des valeurs d'un ou plusieurs attributs d'un élément existant;
- affichage des valeurs d'un ou plusieurs attributs d'un élément existant.

On note que seulement les trois premiers types d'opération entraînent un changement de la banque de données. Pour ces types-là, l'opération est accomplie de la façon suivante:

1) Toute l'information nécessaire pour l'opération est accumulée.
2) L'indicateur "modification en cours" est allumé et écrit dans la banque de données.
3) Tous les changements dans la banque de données nécessaires pour réaliser l'opération sont effectués.

4) L'indicateur "modification en cours" est éteint et écrit dans la banque de données.

5) Le curseur graphique réapparaît, prêt à accepter la prochaine commande. Ceci indique que l'opération a été complétée et que la banque de données se retrouve dans un état consistant.

Cette méthode a été choisie pour sa simplicité et sa robustesse, même si elle n'essaie point de minimiser le nombre d'accès-disques global, i.e. au niveau de l'ensemble de toutes les opérations.

Présentement, EMME/2 comprend trois modules pour l'édition interactive-graphique de la banque de données:

module 1.41 - lignes de démarcations

module 2.12 - réseau de base (centroïdes, noeuds et liens)

module 2.22 - lignes de transport en commun.

Des descriptions détaillées de ces modules se trouvent dans la section IV du manuel de l'usager de EMME/2. Nous nous limitons ici à donner un aperçu général du concept d'une feuille de travail.

La figure III-4-5 montre un exemple d'une feuille de travail vierge. L'exemple provient du module 2.12, qui permet l'édition du réseau de base. La feuille est divisée en quatre parties, dénotées de "a" à "d" dans la figure III-4-5 et décrites ci-bas:

A) La marge à gauche de l'écran sert pour l'*entrée* et la *sortie* de l'information *alphanumérique*. Une grande partie des manipulations peut être accomplie d'une façon purement graphique, mais il reste toujours de l'information qui doit être traitée de façon alphanumérique. Pour chaque opération, une ligne de texte est générée qui contient le type

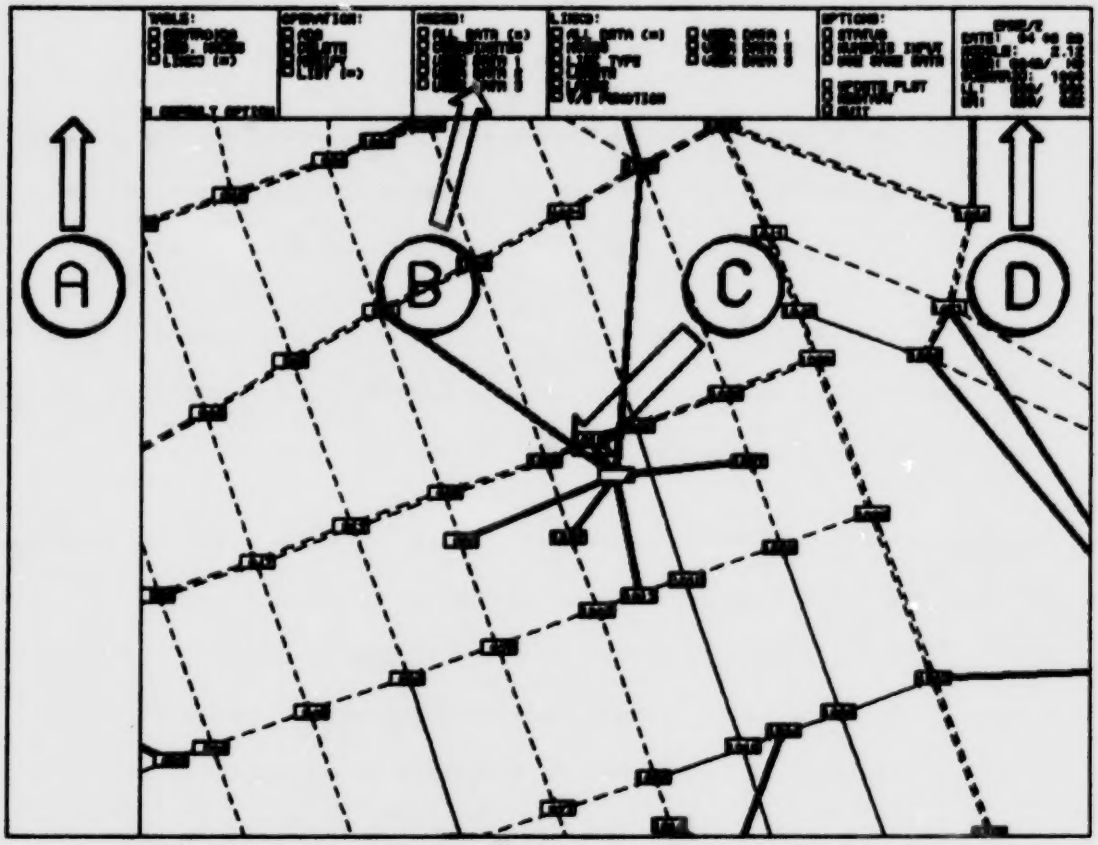

FIGURE III-4-5: Format général d'une feuille de travail.

d'opération (A="add", D="delete", M="modify" et L="list") ainsi que l'identificateur de l'élément en question. Selon le type d'opération, cette ligne est suivie d'autres, une par attribut affecté par l'opération. En cas d'un ajout, l'usager est requis de spécifier la valeur de l'attribut. Pour une modification, sa valeur courante est affichée et l'usager peut, s'il le désire, la changer. Pour l'opération d'affichage, finalement, la ligne montre la valeur courante de l'attribut. A la fin de la session, cette partie constitue donc un compte rendu des opérations effectuées. Par convention, tous les champs qui suivent des deux-points (":") correspondent à des valeurs d'attributs affichées par le système, tandis que tous les champs qui suivent un signe d'égalité ("=") sont des valeurs d'attributs qui ont

été entrées par l'usager. Si toute la marge gauche est remplie avant que l'usager n'ait demandé une nouvelle feuille, la partie d) de la feuille sert comme aire de débordement. A cause de l'espace restreint disponible sur l'écran, les messages d'erreur sont limités à trois sons de cloche, accompagnés d'un numéro d'erreur. Si nécessaire, l'usager peut trouver une explication détaillée de l'erreur dans l'appendice A du manuel de l'usager.

B) La partie du haut de l'écran contient les *commandes*. Chaque commande porte un nom qui est affiché à coté d'un carré. Pour activer une commande, l'usager pointe l'un de ces carrés à l'aide du dispositif d'entrée graphique. Les commandes sont regroupées en blocs logiques et les blocs sont disposés hiérarchiquement de gauche à droite. Chaque feuille de travail a un jeu de commandes différentes, sauf pour les commandes spécifiques à la gestion de la feuille de travail, qui sont toujours disponibles. Voici leurs noms et fonctions:

STATUS: Les commandes présentement actives sont indiquées par la mise en évidence des carrés correspondants.

NUMERIC INPUT: Au lieu d'identifier ou de localiser graphiquement un élément, l'usager peut appeler cette commande. Elle lui permet d'entrer numériquement l'identificateur de l'élément ou la position désirée. Cette option permet, entre autres, d'effectuer des opérations sur des éléments dont ont ne connaît que le nom (mais pas leur position sur l'écran), ou qui ne se trouvent pas dans la fenêtre affichée.

USE SAME DATA: Les valeurs qui ont été spécifiées pour l'opération précédente seront utilisées pour toutes les opérations subséquentes, jusqu'au prochain changement d'opération.

UPDATE PLOT: Une nouvelle feuille de travail est générée qui incorpore toutes les modifications faites jusqu'ici. Cette commande, si invoquée avec le caractère correspondant, permet aussi l'appel aux commandes graphiques "N", "W" et "1" à "9", telles que décrites ci-haut.

RESTART: On retourne au début du module courant.

QUIT: On termine l'exécution du module courant pour retourner au menu principal.

C) La *partie principale* de l'écran contient l'affichage graphique du réseau. Elle a les mêmes dimensions que celle d'un dessin. C'est ici que les éléments qui sont sujets à une opération sont identifiés en les pointant avec le dispositif d'entrée graphique.

D) Le coin supérieur droit de l'écran contient l'information sur la *date*, le *module*, l'*installation*, l'*usager*, le *scénario* et la *fenêtre* courante.

La figure III-4-6 montre la même feuille de travail qu'on a vue dans la figure III-4-5, après que l'usager ait executé les opérations suivantes:

- affichage des attributs du lien 1020 - 1018;
- élimination des liens 1020 - 1018 et 1018 - 1016;
- affichage des attributs du noeud 1062;

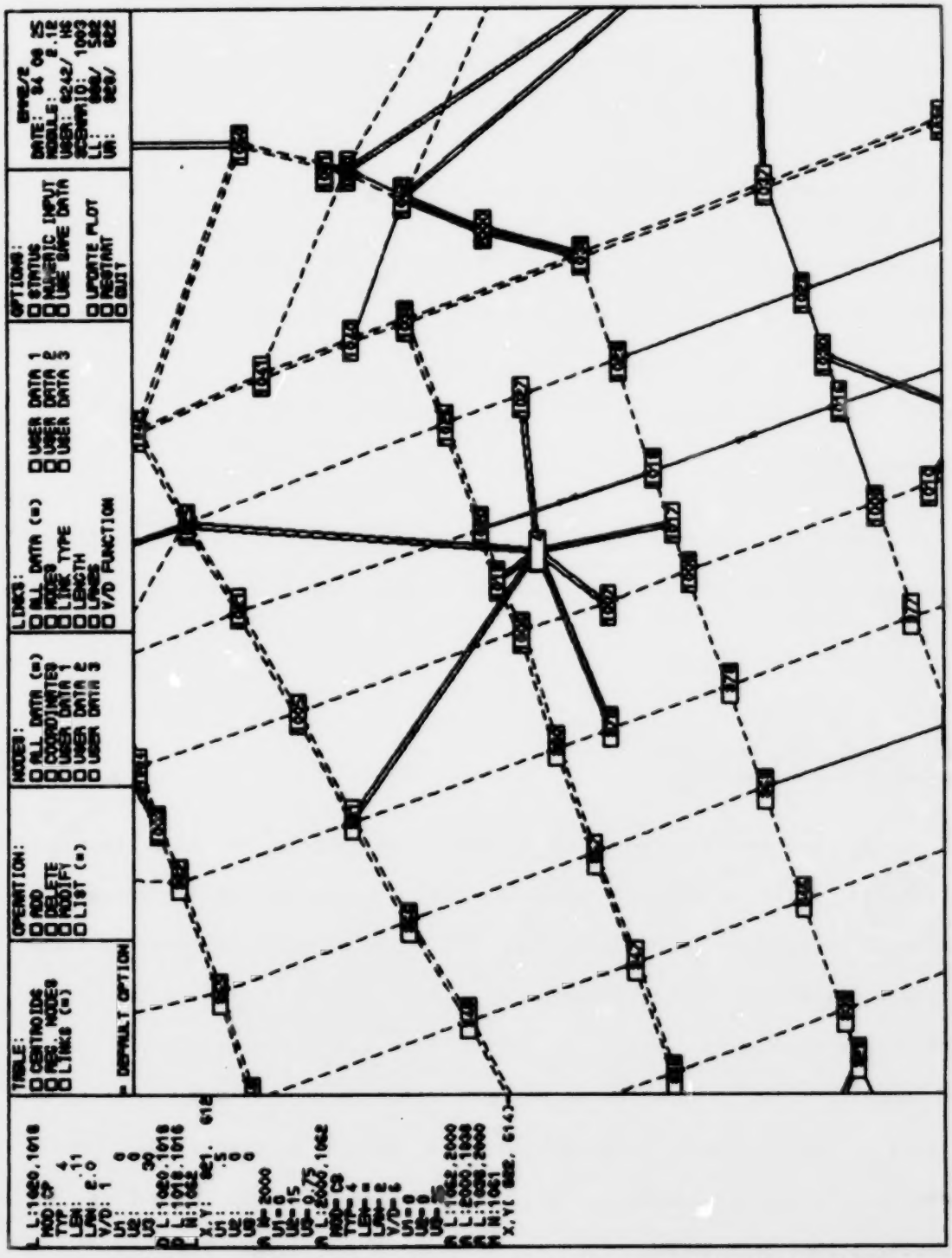

FIGURE III-4-6: Feuille de travail après une suite d'opérations.

- création du noeud 2000;

- création des liens qui relient le noeud 2000 aux noeuds voisins 1038 et 1062 dans les deux sens;

- changement des coordonnées du noeud 1061.

La marge gauche de la figure III-4-6 montre le compte rendu de ces opérations. Dans la partie principale, les liens éliminés sont rayés, de même l'ancienne boite du noeud 1061. Les liens incidents au noeud 1061 n'ont pas changé de place. Ceci illustre donc le caractère "brouillon" de la feuille de travail.

En utilisant la commande "UPDATE PLOT", une nouvelle feuille peut être générée. Elle reflète les modifications ci-haut, comme le montre la figure III-4-7.

4.6 Le rôle des couleurs

Dans la première année du développement de EMME/2, un seul terminal graphique était disponible au Centre de Recherche sur les Transports, un terminal monochrome TEKTRONIX 4014-1. En vue du progrès rapide de la technologie des équipements informatiques, on prévoyait déjà que l'introduction des écrans à couleurs n'était pas loin. Pour que le projet EMME/2 puisse en profiter dès qu'ils deviendraient disponibles, il a fallu intégrer l'aspect "couleur" dans la conception de la partie graphique de EMME/2, même avant d'avoir accès à un tel équipement et de connaître ses spécifications techniques.

Comme on l'a vu dans la section III-4-2, le système graphique de base,

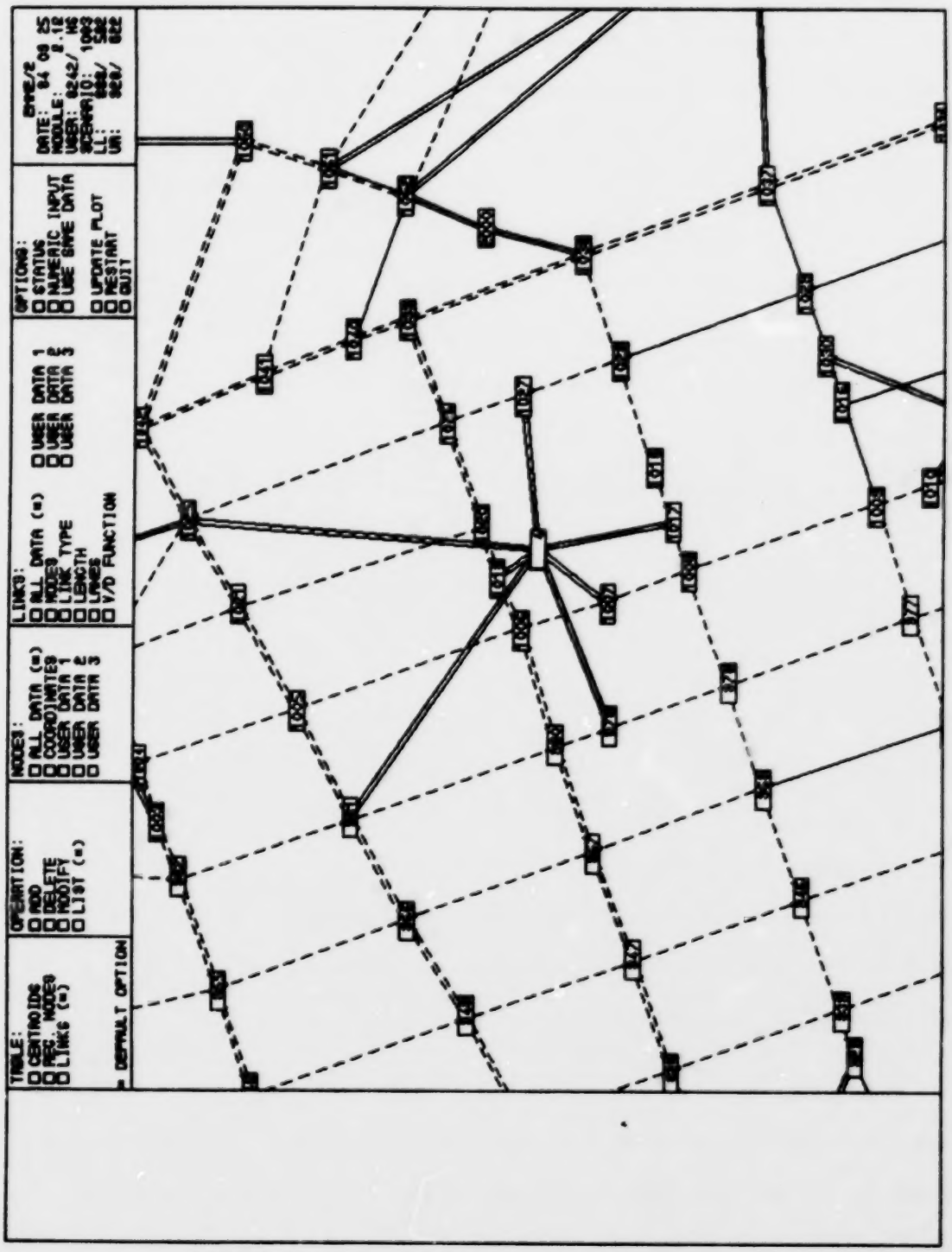

FIGURE III-4-7: Feuille de travail regénérée.

QPLOT, inclut une primitive pour changer la couleur d'affichage. Le traitement des couleurs par cette primitive est très simple mais en même temps assez général pour qu'il puisse s'appliquer à tout type de terminal graphique, y inclus les terminaux monochromes. Au niveau de l'appel à la primitive, un nombre illimité (*) de couleurs est disponible au programmeur. Les couleurs sont identifiées par un *indice de couleur*, i.e. un entier non-négatif. L'indice 0 est réservé pour la *couleur de fond* et l'indice 1 correspond à la *couleur principale*. Tous les indices plus grands sont appelés des *couleurs alternatives*. Elles sont, par définition, différentes de la couleur de fond et, si l'équipement le permet, différentes de la couleur principale. Puisque, pour un équipement donné, seulement un nombre fini de couleurs distinctes est possible, il n'y a qu'un nombre fini de couleurs alternatives différentes, qui sont répétées cycliquement.

On note que le système ne connaît pas la couleur affichée, mais seulement un indice de couleur. Il est donc impossible de demander au système de dessiner un élément en "vert pâle" et l'autre en "écarlate". La correspondance entre l'indice de couleur et la couleur affichée à l'écran est dépendante de l'équipement utilisé. Par convention, si l'équipement le permet, les trois premières couleurs alternatives (indices 2, 3, et 4) correspondent aux couleurs de base "rouge", "vert" et "bleu".

Même si aujourd'hui les écrans polychromes sont plus répandus, la plupart des usagers de EMME/2 utilisent encore, et ont l'intention de continuer à utiliser, des terminaux monochromes. Les raisons en sont multi-

(*) Pour être plus précis, ce nombre n'est limité que par l'entier le plus grand que peut traiter l'ordinateur.

ples, voici les plus importantes:

-coût élevé des terminaux graphiques polychromes,

-résolution plus faible des écrans à couleur,

-le coût de reproduction d'un dessin en couleurs est prohibitif pour la plupart des applications.

C'est donc pour ces raisons que les *couleurs* ne jouent qu'un *rôle auxiliaire* dans l'affichage graphique de EMME/2. Elles servent à rendre le dessin *plus lisible*, en dessinant différentes composantes du dessin en couleurs différentes. Les couleurs elles-mêmes, par contre, ne contiennent pas d'information supplémentaire.

Dans l'implantation actuelle de EMME/2, l'usager n'a pas la possibilité de spécifier les indices de couleur qui sont utilisées pour les différents éléments; c'est le système qui les affecte automatiquement. Cette approche a été choisie pour que l'implantation des couleurs soit transparente à un usager qui ne les utilise pas. La décision devra être reconsidérée au moment où l'utilisation des écrans couleurs sera devenue plus répandue. Dans ce cas, il suffirait d'ajouter à la banque de données l'attribut "indice de couleur" pour chaque élément.

Jusqu'à maintenant, EMME/2 a été porté sur deux types d'écran polychrome: La station de travail évoluée CHROMATICS modèle CGC 7900 et le terminal graphique de haut de gamme TEKTRONIX 4115B. Il est intéressant de noter que, dans les deux cas, le travail d'adaptation des routines "driver" de GPLOT n'a pas pris plus de deux journées de travail.

5. Les modules de EMME/2

Comme nous l'avons déjà mentionné, le système EMME/2 est composé d'environ 40 modules différents. Dans cette section, nous examinerons les modules les plus importants. Nous donnerons aussi les particularités de l'implantation, lorsque cela s'avérera intéressant. Cependant, il est à noter que cette section ne constitue pas un manuel d'usager des modules. Notre but est plutôt de donner un aperçu général des modules et de leur fonctionnement. Des descriptions plus complètes peuvent être retrouvées dans le chapitre IV du manuel de l'usager de EMME/2, qui contient tous les renseignements nécessaires pour l'utilisation des modules.

Le tableau III-5-1 montre une liste de tous les modules présentement disponibles dans EMME/2. Quelques autres modules, déjà conçus logiquement, mais pas encore programmés, s'ajouteront sous peu à cette liste. Comme on le voit dans le tableau III-5-1, les modules sont partitionnés en six groupes. Chaque groupe de modules s'occupe d'un certain aspect du système. Le groupe 1 contient les modules utilitaires, i.e. les modules qui sont reliés plus à la gestion de la banque de données et de l'environnement de travail qu'aux problèmes de planification de transport. Les groupes 2, 3 et 4 sont les éditeurs des trois principales composantes de données: réseau, matrices et fonctions. Le terme "éditeur" englobe tous les outils nécessaires à l'entrée, la modification et la sortie graphique et alphanumérique des éléments correspondants. Le groupe 5 s'occupe des affectations et le groupe 6 comprend les modules qui permettent l'analyse des résultats d'affectation.

1.	LES UTILITAIRES
1.11	Manipulations directes de la banque de données
1.21	Consultation du journal de bord
1.22	Manipulations de scénarios
1.23	Titres et unités
1.41	Entrée /modification / affichage des démarcations
2.	L'EDITEUR DE RESEAU
2.01	Entrée / modification / sortie des modes
2.02	Entrée / modification / sortie des véhicules de t.c.
2.11	Entrée séquentielle du réseau de base
2.12	Modification interactive - graphique du réseau de base
2.13	Affichage du réseau de base
2.14	Sortie alphanumérique du réseau de base
2.15	Affichage des plus courts chemins sur le réseau de base
2.21	Entrée séquentielle des lignes de transport en commun
2.22	Modification interactive - graphique des lignes de t.c.
2.23	Affichage des lignes de transport en commun
2.24	Sortie alphanumérique des lignes de transport en commun
2.31	Entrée / modification / affichage des virages pénalisés
3.	L'EDITEUR DE MATRICES
3.01	Entrée / modification / sortie des groupes de zones
3.11	Entrée séquentielle de matrices
3.12	Entrée / modification interactive de matrices
3.13	Affichage graphique de matrices
3.14	Sortie alphanumérique de matrices
3.16	Affichage d'histogrammes
3.21	Calculatrice interactive de matrices
3.22	Procédures de balancement de matrices
4.	L'EDITEUR DE FONCTIONS
4.11	Entrée séquentielle de fonctions
4.12	Entrée / modification interactive de fonctions
4.13	Affichage graphique de fonctions
4.14	Sortie alphanumérique et tabulation de fonctions
5.	LES PROCEDURES D'AFFECTATION
5.11	Préparation d'un scénario pour une affectation
5.21	Affectation - Auto
5.31	Affectation - Transport en commun
6.	L'ANALYSE DES RESULTATS
6.11	Rapport de temps et volumes-auto
6.12	Affichage de temps et volumes-auto
6.13	Comparaison graphique de temps et volumes-auto
6.14	Affichage des volumes sur les intersections
6.15	Accessibilité et plus courts chemins sur le réseau-auto
6.21	Rapport de temps et volumes - transport en commun
6.22	Affichage de temps et volumes - transport en commun

TABLEAU III-5-1: Les modules de EMME/2.

Dans les pages suivantes, nous présenterons brièvement le fonctionnement de chaque module. Les modules graphiques seront illustrés, à titre d'exemple, par des affichages provenant du réseau de la ville de Winnipeg (Manitoba). Il s'agit d'un réseau de taille moyenne, comportant environ 150 zones origines / destinations, 1200 noeuds, 3000 liens, 70 lignes de transport en commun et 4500 segments de ligne.

5.1 Les utilitaires

On a regroupé ici tous les modules servant surtout à la gestion du système et de la banque de données.

Le module 1.11, "Manipulations directes de la banque de données", est un outil de débogage (en anglais "debugging"), qui, dans des circonstances normales, ne devrait jamais être appelé par l'usager. Ce module permet d'accéder à n'importe quel mot dans la banque de données et de le modifier directement. Il va de soi que l'utilisation de ce module devrait être réservée à des personnes ayant une connaissance approfondie des structures de données utilisées, puisqu'une petite erreur dans son utilisation peut rendre inutilisable la banque au complet. Si utilisé avec prudence et compétence, ce module sert à analyser les problèmes "inexplicables" et, si possible, à prendre des mesures de sauvetage.

Le module 1.21, "Consultation du journal de bord", permet l'affichage sélectif de l'histoire récente de la banque de données. Le journal de bord contient à tout moment l'information pertinente sur un certain nombre d'appels de modules. Le nombre d'entrées, typiquement entre 300 et 500, dépend de la longueur des commentaires associés aux entrées. A l'aide de

ce module, l'usager peut sélectionner les entrées qui l'intéressent, selon les critères suivants:

- la date de l'appel du module
- le numéro du module
- les initiales de l'usager
- le scénario courant

Pour chaque critère, l'usager peut spécifier les valeurs à inclure dans le rapport, ou, alternativement, les valeurs à exclure. Le rapport généré contient toutes les entrées qui satisfont les critères spécifiés, en commençant par l'entrée la plus récente.

Le module 1.22, "Manipulation de scénarios", permet de créer, d'éliminer et de copier des scénarios. Chacune de ces opérations entraîne une entrée automatique dans le journal de bord. Puisque l'opération de copier le contenu d'un scéario dans un autre implique la lecture et l'écriture d'un ensemble de données d'une taille assez grande, cette opération n'est pas instantanée, mais requiert généralement une à deux minutes. A l'intérieur du même module, l'usager peut aussi changer les indicateurs de protection du scénario courant.

Le module 1.23, "Titres et unités", permet à l'usager de redéfinir le titre du projet et la description du scénario courant, ainsi que de choisir les unités utilisées. Le système EMME/2 connaît trois unités qui peuvent être définies par l'usager:

- l'unité de distance (km)
- l'unité de coût ($)
- l'unité de consommation d'énergie (MJ)

Les valeurs par défaut sont indiquées entre parenthèses. Puisque les

attributs qui sont mesurés dans ces unités ne sont jamais sujets à des conversions, le choix de l'unité consiste simplement à spécifier un nom.

Le module 1.41, "Entrée / modification / affichage des démarcations", s'occupe de la gestion des démarcations. La partie principale du module est une feuille de travail qui permet à l'usager l'entrée et la modification interactive graphique des démarcations. La figure III-5-1 montre un exemple de cette feuille de travail sur laquelle l'usager vient de définir les trois pistes de l'aéroport comme une démarcation. Les deux rivières, dessinées avec le symbole ">", sont, elles aussi, des démarcations. Dans la partie des commandes de la feuille, on reconnaît la matrice des paramètres, où l'usager choisit, à l'aide du dispositif d'entrée graphique, le symbole utilisé, sa taille, l'intervalle et le type d'interpolation.

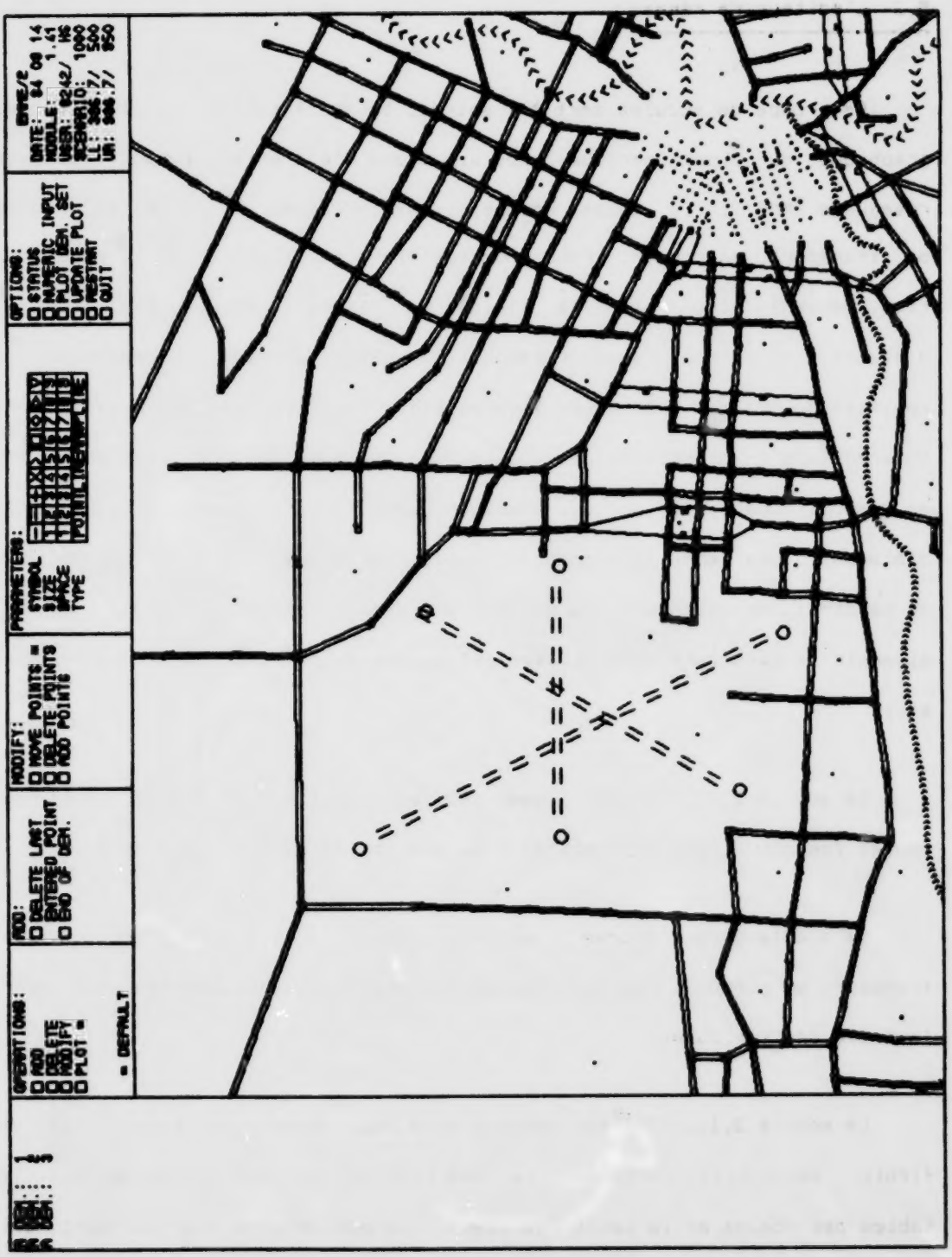

FIGURE III-5-1: Entrée interactive-graphique des délarcations.

5.2 L'éditeur de réseau

Ce groupe de modules sert à l'entrée, la modification et la sortie, graphique ou alphanumérique, des six types d'éléments qui constituent le réseau de EMME/2, i.e. modes, noeuds, liens, virages, véhicules et lignes de transport en commun. Pour tous ces éléments, l'entrée et la modification peuvent être faites à l'aide de fichiers séquentiels préparés d'avance, ou d'une façon interactive. Deux modules incorporent des feuilles de travail graphiques, permettant l'entrée et la modification interactives - graphiques des noeuds, des liens et des lignes de transport en commun. Quant à la sortie, l'usager peut générer, pour chaque type d'élément, des rapports alphanumériques, destinés soit à l'imprimante soit au terminal, des fichiers séquentiels en format images-cartes, et, pour les éléments à caractère spatial, des affichages graphiques sous forme de dessin.

Le module 2.01, "Entrée / modification / sortie des modes", regroupe toutes les opérations nécessaires à la gestion de la table des modes.

Le module 2.02, "Entrée / modification / sortie des véhicules de transport en commun", regroupe toutes les opérations nécessaires pour gérer la table des véhicules.

Le module 2.11, "Entrée séquentielle du réseau de base", lit un fichier séquentiel contenant la description du réseau de base, i.e. la tables des noeuds et la table des liens. Le module génère un rapport qui répète l'information lue et contient tous les messages d'erreurs, s'il y en a.

Le module 2.12, "Entrée / modification interactive - graphique du réseau de base", permet à l'usager de construire ou de modifier le réseau de base à l'aide d'une feuille de travail graphique. Toutes les opérations se font au niveau d'un seul élément, i.e. un noeud ou un lien. Puisqu'un changement dans la table des centroïdes a de graves répercussions sur les matrices, les opérations sur les noeuds sont strictement séparées entre celles sur les centroïdes et celles sur les noeuds réguliers. La figure III-5-2 montre un exemple d'une telle feuille de travail. Des exemples plus complets peuvent être trouvés dans la section 4.5 de ce chapitre, où le module 2.12 a été utilisé pour illustrer le concept de la feuille de travail.

Le module 2.13, "Affichage du réseau de base", permet de dessiner le réseau de base, soit en entier, soit sous forme de sous-réseau logique ou géographique. Le dessin peut contenir, optionnellement, les valeurs des attributs de lien ou de noeud. Les figures III-5-3 et III-5-4 donnent des exemples de dessins générés par ce module.

Le module 2.14, "Sortie alphanumérique du réseau de base", génère des rapports et / ou des tables séquentielles contenant la description des noeuds et des liens. De plus, ce module peut générer un sommaire du réseau de base, en regroupant l'information par mode, type de lien et fonction de volume-délai.

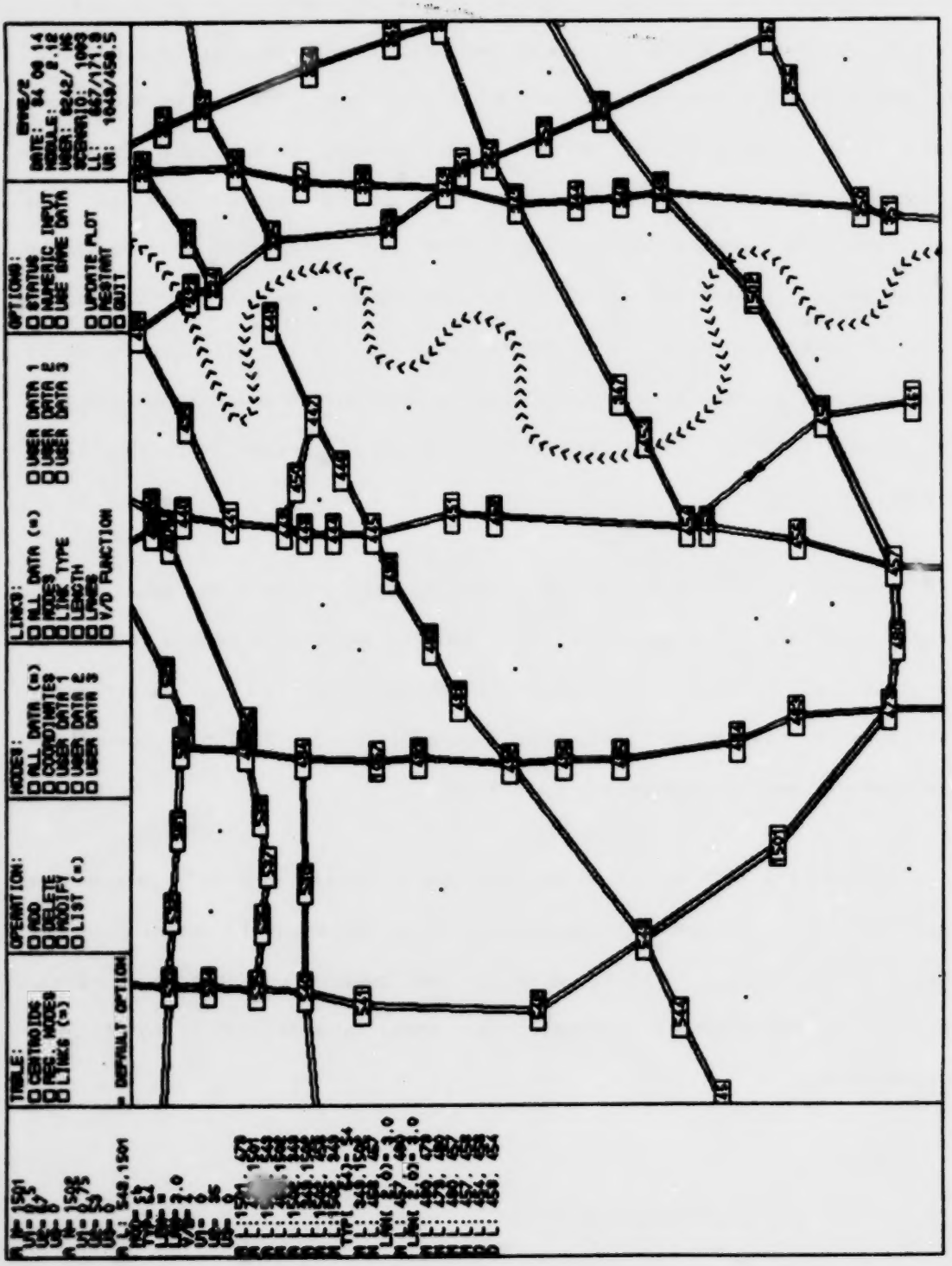

FIGURE III-5-2: Modification interactive-graphique du réseau de base.

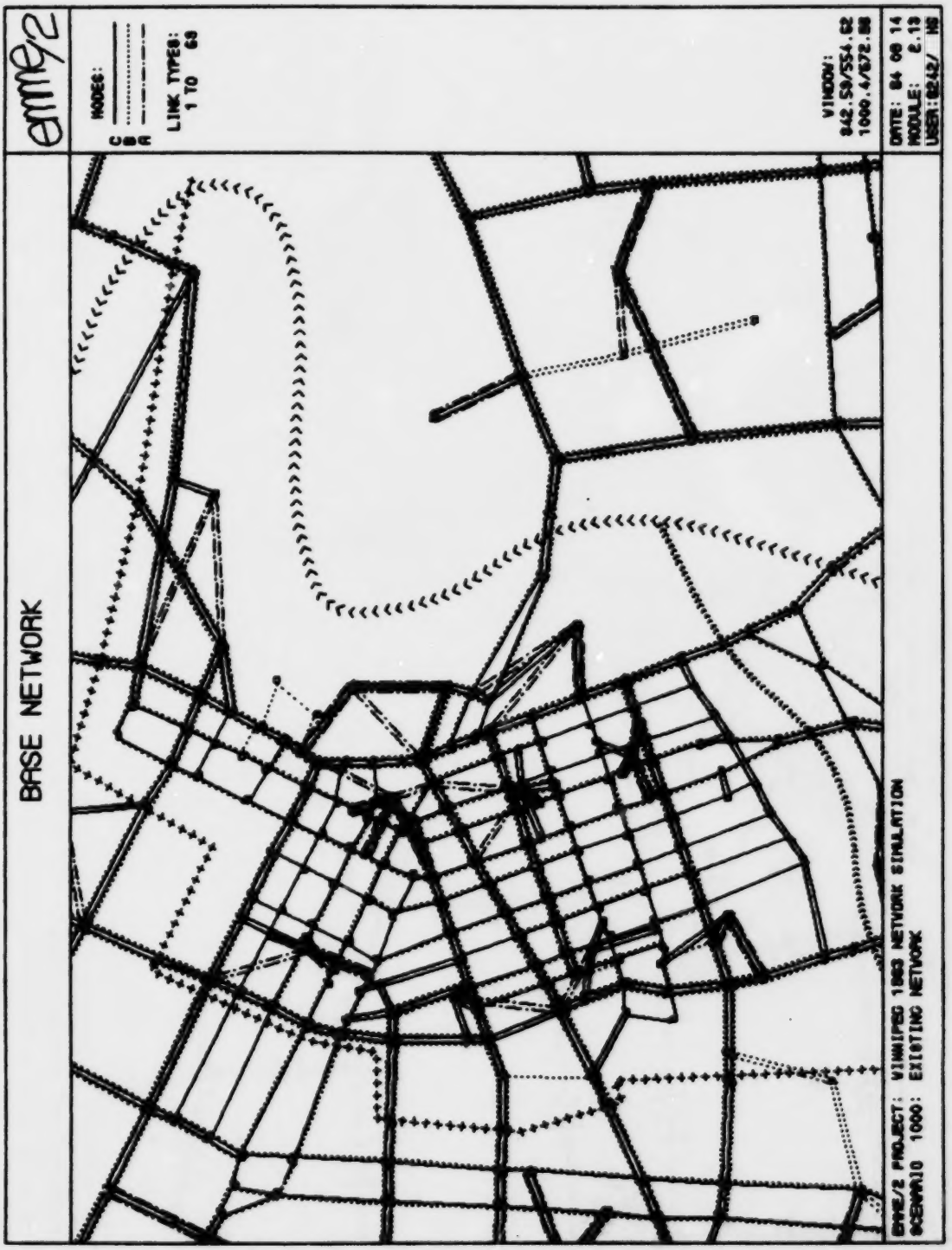

FIGURE III-5-3: Réseau de base - modes permis sur les liens.

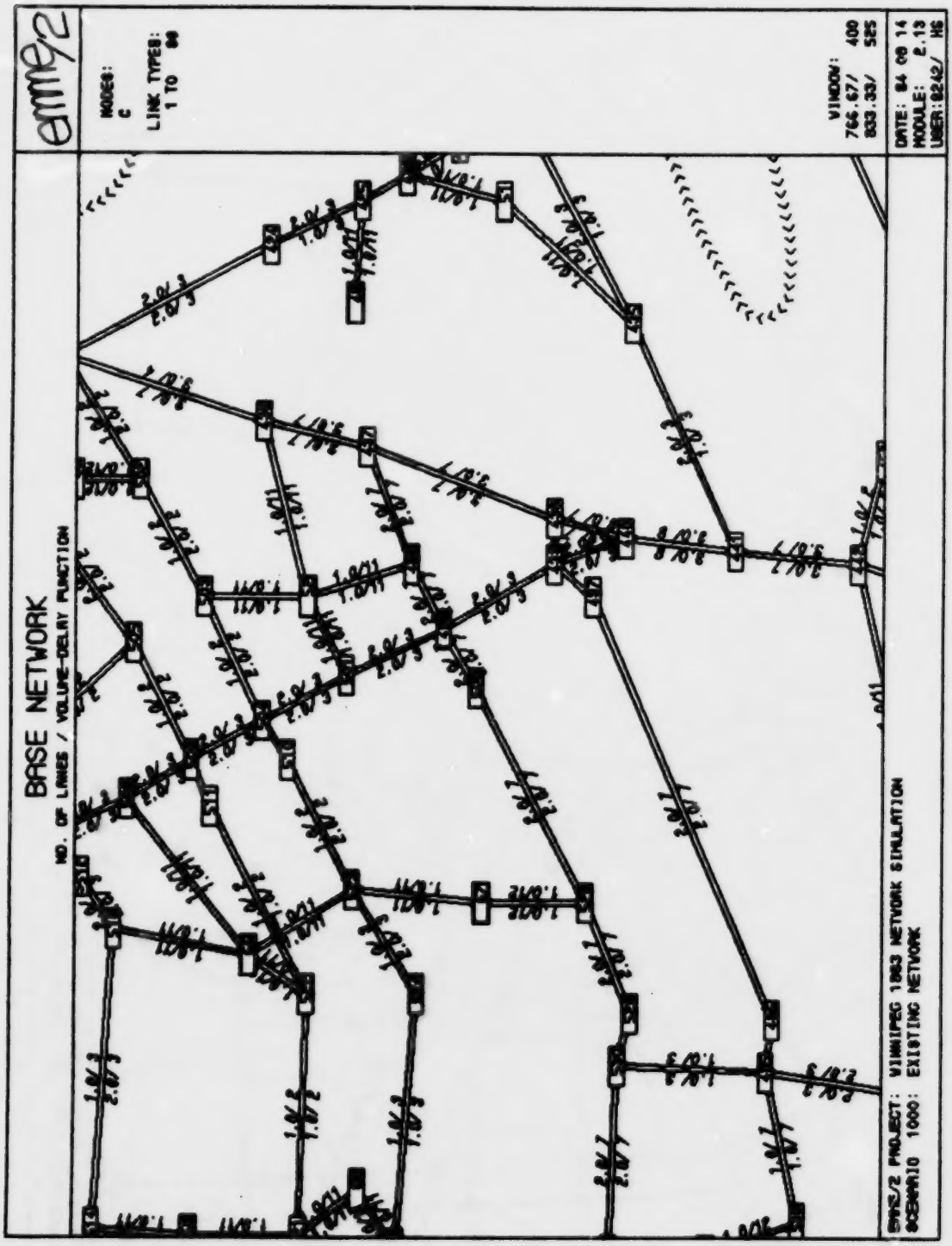

FIGURE III-5-4: Réseau de base - nb. de voies et fonction volume-délai.

Le module 2.15, "Affichage des plus courts chemins sur le réseau de base", est un outil polyvalent, qui sert aussi bien à la validation des données du réseau qu'à répondre à toute une gamme de questions intéressantes, reliées aux chemins et distances sur le réseau de base. Les chemins sont calculés sur un réseau partiel donné par un sous-ensemble de modes, en fonction de la distance ou d'une des trois données de l'usager sur les liens. Le module traite les chemins individuels, les arbres et les forêts. Un mode d'enquête interactive - graphique permet à l'usager d'indiquer des paires de noeuds, qui sont reliés instantanément par le chemin le plus court, et la longueur totale du chemin est indiquée sur la marge droite du dessin. Le module utilise un algorithme de type Dijkstra (1959) avec des étiquettes sur les noeuds. Une structure de tas est employée pour gérer les étiquettes de noeud - permettant une évaluation très rapide des chemins. La figure III-5-5 montre une utilisation de ce module qui n'est pas reliée directement à la planification de réseaux de transports. L'exemple montre une forêt de plus courts chemins, comportant un arbre pour chaque hôpital de la ville. Pour chaque noeud du réseau, la forêt contient le chemin vers l'hôpital le plus proche. Notons qu'en utilisant un terminal à écran-couleur, chaque arbre de la forêt est dessiné en une couleur différente. Dans le deuxième exemple, contenu dans la figure III-5-6, le chemin le plus court, en terme de distance bien entendu, est affiché. Sa longueur totale est de 9.94 milles.

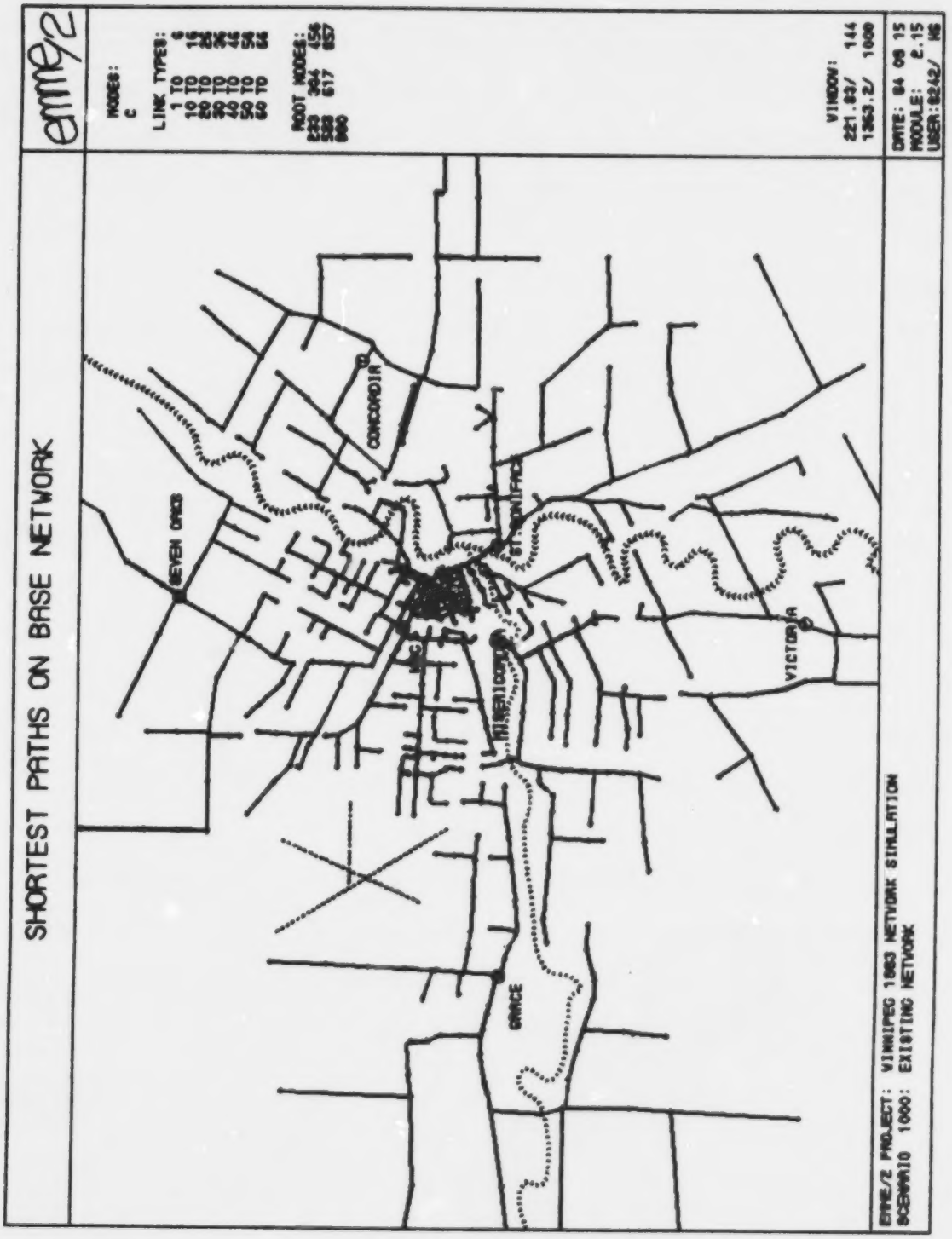

FIGURE III-5-5: Le plus court chemin vers l'hôpital le plus proche.

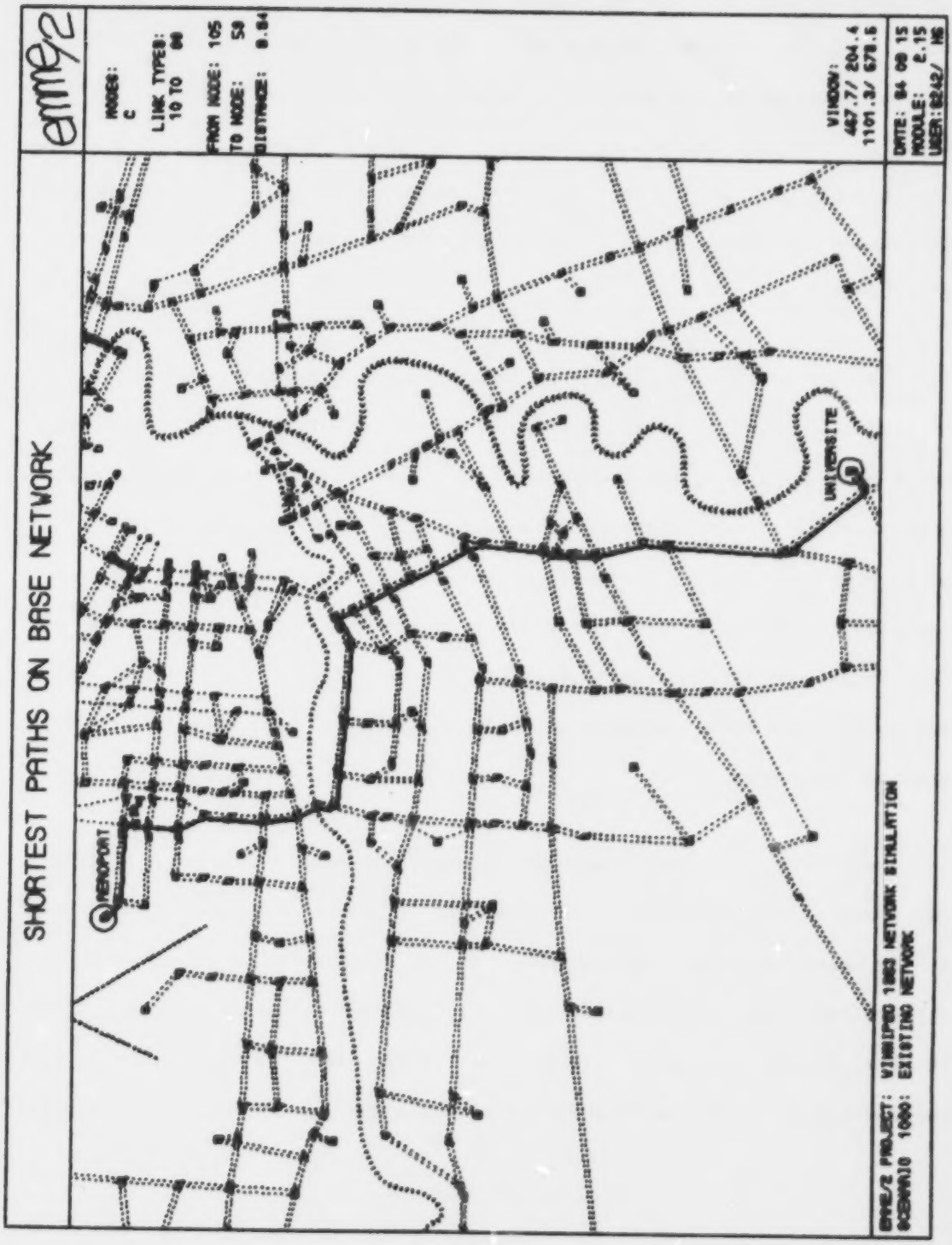

FIGURE III-5-6: Le plus court chemin de l'université à l'aéroport.

Le module 2.21, "Entrée séquentielle des lignes de transport en commun", lit un fichier séquentiel contenant la description d'une ou de plusieurs lignes de transport en commun. L'itinéraire d'une ligne est spécifié par une suite de noeuds. Entre chaque paire de noeuds, s'il n'existe pas de lien direct, le plus court chemin sur le réseau qui correspond au mode de la ligne est calculé et est utilisé pour compléter l'itinéraire.

Le module 2.22, "Modification interactive - graphique des lignes de transport en commun", permet à l'usager de définir de nouvelles lignes et / ou de modifier les lignes existantes. Ceci est fait en mode interactif - graphique, à l'aide d'une feuille de travail. L'usager définit une ligne en pointant sur une suite de noeuds, qui sont reliés automatiquement par les plus courts chemins (en distance) pour former l'itinéraire de la ligne. Ceci permet une interaction très efficace avec l'usager; une ligne, comme celle contenue dans l'exemple de la figure III-5-7, peut être définie avec très peu de noeuds.

Le module 2.23, "Affichage des lignes de transport en commun", génère des dessins contenant les itinéraires des lignes ou leurs caractéristiques sur le réseau de base. Le problème de la représentation graphique des itinéraires est particulièrement intéressant puisque, souvent, les itinéraires de plusieurs lignes utilisent les mêmes liens. La représentation graphique doit donc être telle que, si on affiche plusieurs

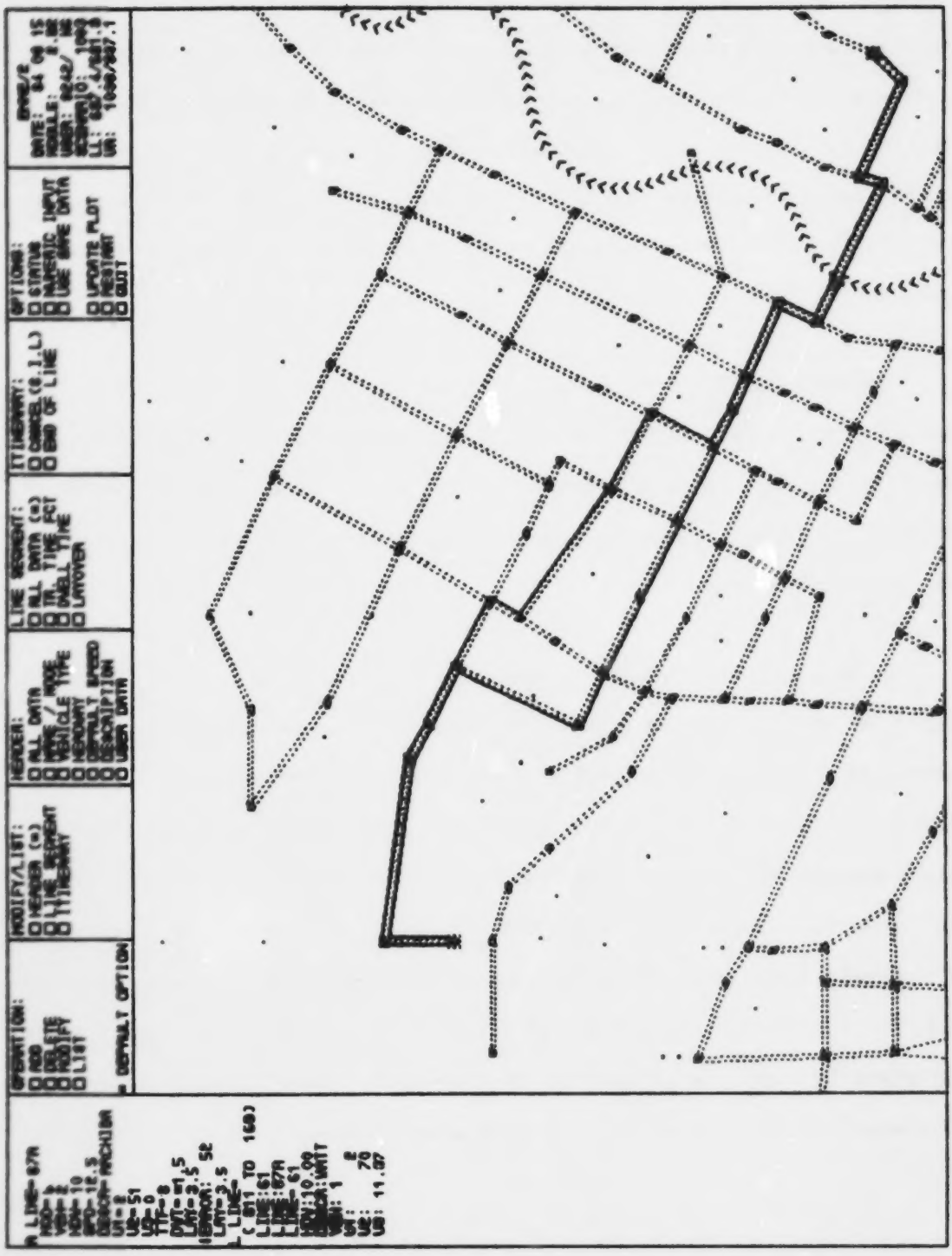

FIGURE III-5-7: Modification interactive - graphique des lignes.

itinéraires sur le même dessin, on puisse suivre chaque itinéraire même là où plusieurs lignes courent en parallèle. Dans ce but, un mécanisme assez complexe a été développé, qui décale graphiquement les itinéraires pour que chacun puisse être suivi individuellement. Un exemple d'un tel dessin est donné dans la figure III-5-8. Les dessins d'itinéraires profitent particulièrement de l'utilisation d'un terminal couleurs, puisque, dans ce cas, chaque ligne est illustrée par une couleur différente.

Le module 2.24, "Sortie alphanumérique des lignes de transport en commun", génère des rapports et / ou des tables séquentielles contenant les descriptions des lignes de transport en commun. Aussi, ce module permet de générer un rapport sommaire, résumant les caractéristiques les plus importantes de chaque ligne.

Le module 2.31, "Entrée / modification / affichage des virages", regroupe toutes les opérations associées aux intersections et aux virages pénalisés. L'entrée et la modification des virages peuvent être faites par fichier séquentiel ou en mode interactif. Les définitions des intersections peuvent être sorties comme rapport, table séquentielle ou sous forme de dessin. La figure III-5-9 montre un affichage graphique d'intersections. Les virages permis (non-pénalisés) sont indiqués par des flèches continues. Les virages pénalisés sont représentés par des flèches contenant le numéro de la fonction de pénalité. Les virages interdits, finalement, sont indiqués par l'absence d'une flèche.

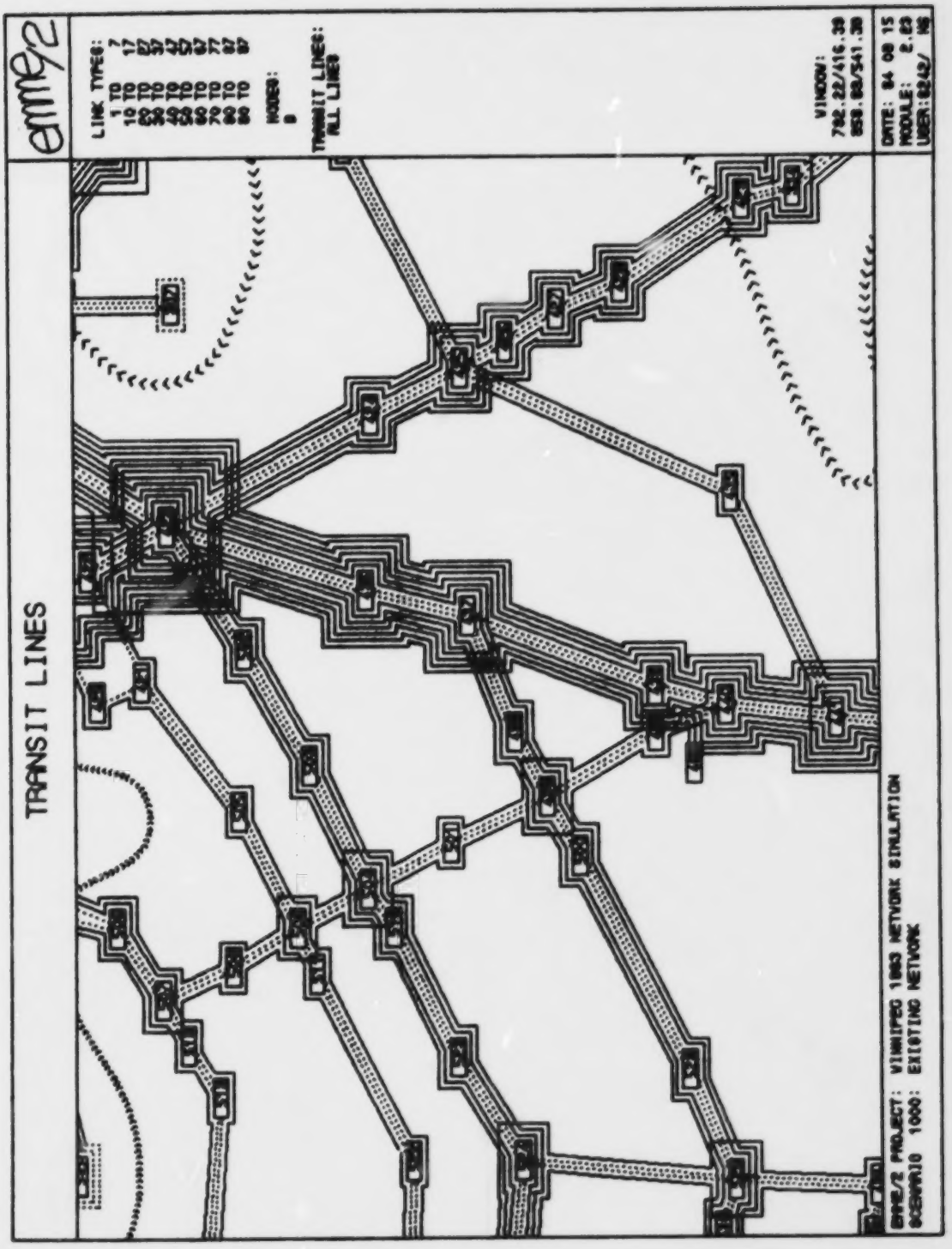

FIGURE III-5-8: L'affichage des itinéraires des lignes.

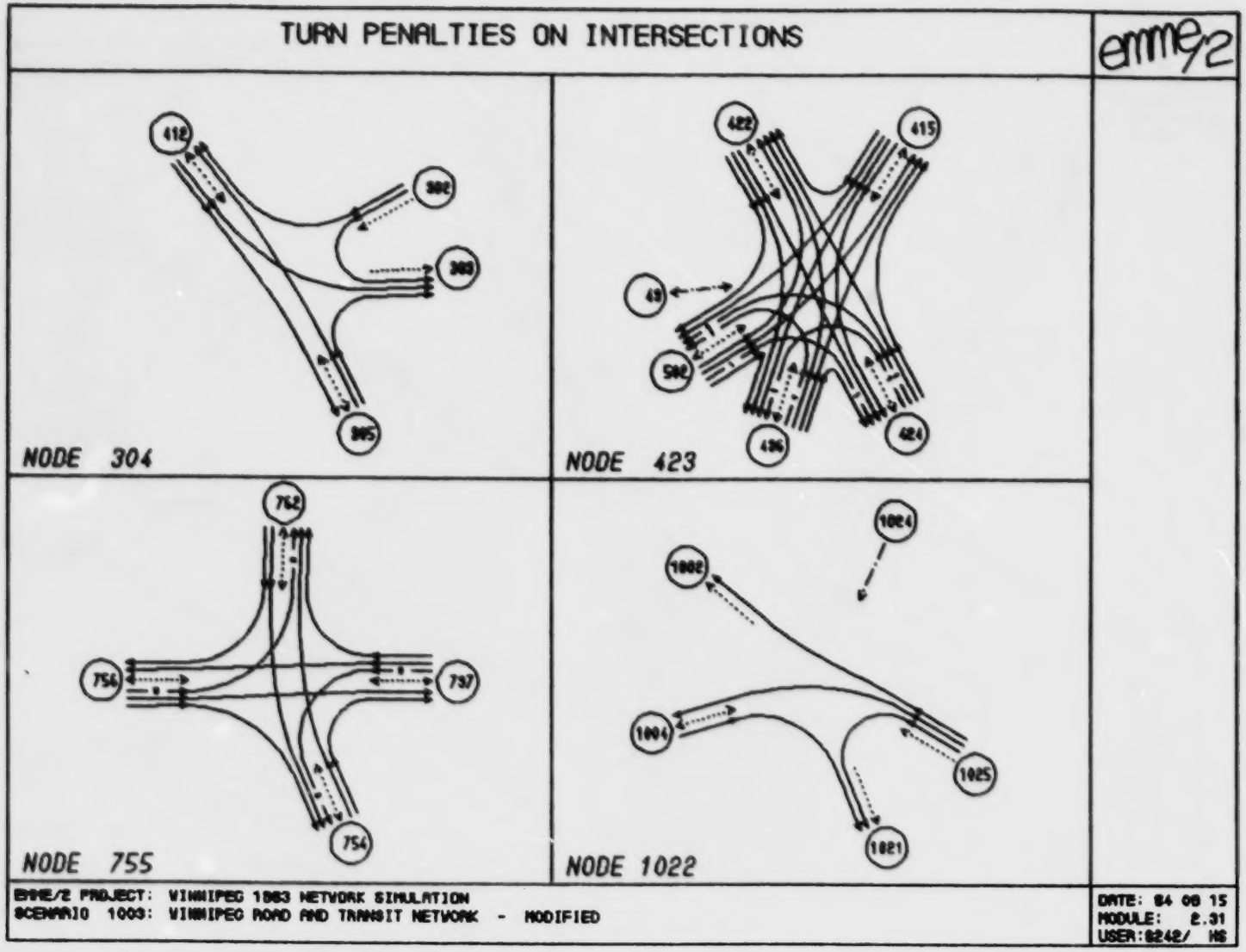

FIGURE III-5-9: Intersections et virages pénalisés.

5.3 L'éditeur de matrices

Dans ce groupe de modules, on trouve tous les outils de gestion, d'affichage et d'analyse des matrices. Les matrices sont traitées d'une façon unifiée, i.e. les modules ignorent le type de données stockées dans une matrice particulière (demande, caractéristique socio-économique, niveau de service, pointeur vers fonction de demande, ...). Cette approche résulte en une gamme d'outils très généraux, qui peuvent, si utilisés avec les données appropriées, réaliser à peu près toutes les opérations traditionnellement effectuées avec des matrice de contenu particulier. Par exemple, l'usager cherchera en vain un module "Affichage de la distribution des temps de voyages"; par contre, le module 3.16, "Affichage d'histogrammes", peut servir pour ce but, mais en même temps, il a beaucoup d'autres utilisations possibles. Les modèles de distribution constituent un autre exemple. Même si le système EMME/2 ne comprend pas de module pour ce but spécifique, la combinaison de deux outils disponibles, la calculatrice de matrice (module 3.21) et les procédures de balancement (module 3.22), permet de réaliser toute une gamme de modèles de distribution.

Le module 3.01, "Entrée / modification / sortie des groupes de zones", regroupe toutes les opérations nécessaires pour traiter les ensembles de groupes de zones. Rappelons-nous que chaque ensemble de groupe constitue une partition des zones origines / destinations selon un critère quelconque. L'usager utilise, dans les autres modules présentés plus bas, les groupes de zones pour la spécification efficace de sous-matrices, pour les entrées et sorties agrégées (modules 3.11 et 3.14) et dans la calculatrice interactive de matrices (module 3.21).

Le module 3.11, "Entrée séquentielle de matrices", lit un fichier séquentiel qui contient une ou plusieurs matrices en format d'images-cartes.

Le module 3.12, "Entrée / modification interactive de matrices", permet à l'usager d'entrer, de copier, d'afficher et de modifier interactivement l'information contenue dans les matrices. Pour chacune de ces opérations, plusieurs formats d'interaction sont possibles; leur énumération serait trop longue et est donc laissée au manuel de l'usager.

Le module 3.13, "Affichage graphique de matrices", permet à l'usager de représenter graphiquement et géographiquement l'information contenue dans une matrice, en affichant les éléments aux centroïdes correspondants. Les données peuvent être affichées graphiquement et / ou numériquement par zone origine, par zone destination ou par paire origine - destination. La figure III-5-10 montre un exemple de *lignes de désir*, i.e. des bandes à largeur proportionnelle à la demande reliant des paires origine - destination. L'exemple montre la demande de chaque zone origine vers les zones du centre-ville pendant l'heure de pointe du matin.

Le module 3.14, "Sortie alphanumérique de matrices", génère des rapports et / ou des tables séquentielles, contenant la description d'une ou de plusieurs matrices. Des sorties agrégées sont possibles en utilisant des ensembles de groupes de zones.

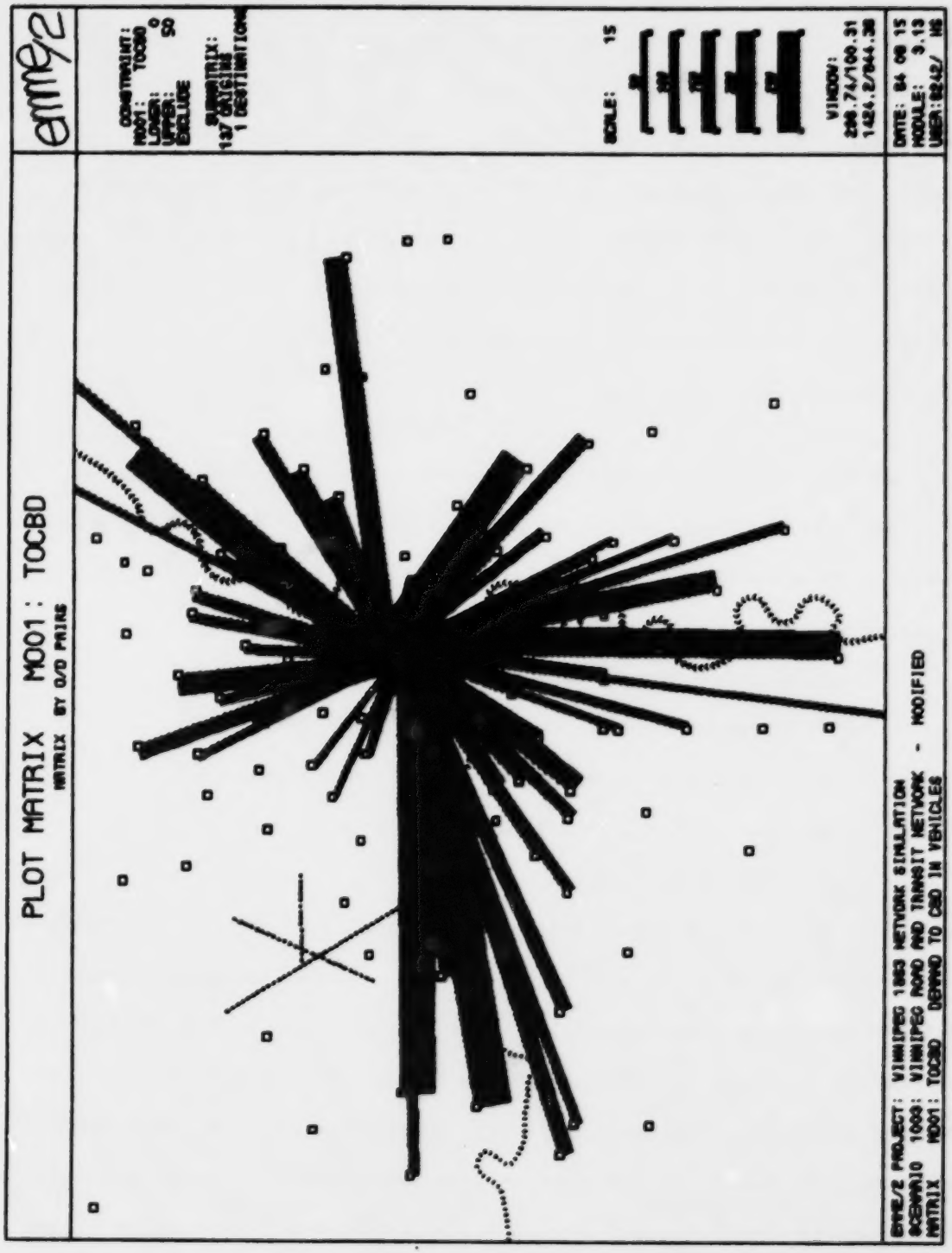

FIGURE III-5-10: La demande vers le centre-ville.

Le module 3.16, "Affichage d'histogrammes", est un puissant outil pour analyser et représenter le contenu des matrices. Un histogramme est basé sur une matrice de données (axe horizontal) et une matrice de poids (axe vertical). Le module peut afficher ou imprimer des histogrammes individuels, des comparaisons et des différences d'histogrammes. Les éléments à inclure dans l'histogramme sont choisis à l'aide des mécanismes de sous-matrice et de matrice de contrainte. La figure III-5-11 montre les distributions de temps de voyages pour les déplacements faits en voiture et les déplacements faits par transport en commun.

Le module 3.21, "Calculatrice interactive de matrices", permet à l'usager de combiner les informations qu'il a déjà stockées dans les matrices existantes, afin de calculer de nouvelles matrices. Dans ce but, l'usager entre une expression algébrique qui peut contenir:

- des identificateurs de matrices existantes
- P, indiquant l'identificateur de la zone origine
- Q, indiquant l'identificateur de la zone destination
- Gx(P) ou Gx(Q), où Gx est un ensemble de groupes de zones
- des constantes numériques

ainsi que tous les opérateurs algébriques, opérateurs logiques et fonctions intrinsèques qu'on a vus dans la section 3.3. Cette expression algébrique est évaluée, élément par élément, résultant en une matrice d'un format qui correspond à l'union de ses différents opérandes. L'usager peut demander de stocker les résultats dans une matrice quelconque, ou, si le résultat n'est qu'un scalaire, il peut être affiché immédiatement. Si une expansion de la matrice résultante est requise, elle est faite automatiquement selon

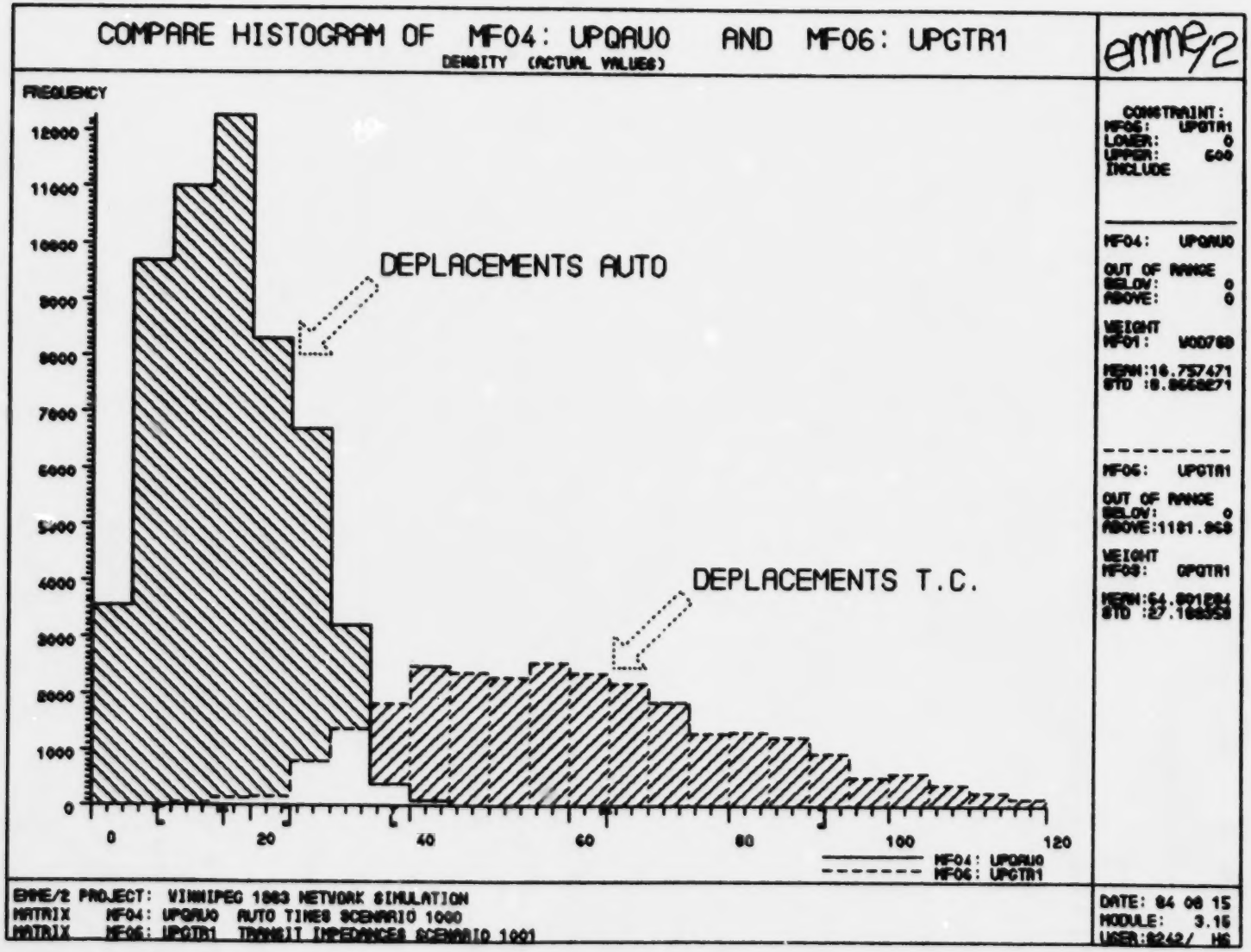

FIGURE III-5-11: Distribution des temps de déplacements.

les règles expliquées dans la section 3.2. Si, par contre, le format de stockage est plus petit que le format de l'expression, une *réduction* est faite avec un opérateur binaire quelconque, spécifié par l'usager (habituellement '+', '.MAX.' ou '.MIN.'). Les calculs ne sont faits que pour le sous-ensemble d'éléments actifs, sélectionnés à l'aide d'une sous-matrice et / ou d'une matrice de contrainte. Illustrons l'utilisation de ce module par l'exemple suivant. Supposons que l'usager ait stocké la demande dans la matrice MF1. Il aimerait maintenant connaître le nombre total de voyages intrazonaux, i.e. la somme des éléments sur la diagonale de la matrice. Pour faire cela, il entre l'expression

```
MS = MF1 * ( P .EQ. Q )
```

où MS indique que le résultat est réduit à un scalaire qui est imprimé immédiatement. Puisque l'expression MF1*(P.EQ.Q) est de type MF (matrice pleine), il faut faire une réduction sur les origines et une sur les destinations. L'usager choisit l'opérateur '+' pour les deux réductions. Le scalaire résultant est, en effet, la somme des éléments sur la diagonale de MF1, i.e. le nombre total de voyages intrazonaux.

Le module 3.22, "Procédures de balancement de matrice", permet à l'usager d'effectuer des balancements proportionnels à deux ou trois dimensions. Le problème de balancement à deux dimensions d'une matrice donnée $\{t_{pq}\}$, aussi connu sous les noms "balancement biproportionnel" , "Méthode de Furness" ou "Fratar", est de trouver les facteurs α_p et β_q, tels que la matrice résultante $\{T_{pq}\}$, donnée par

$$T_{pq} = \alpha_p \beta_q t_{pq} \quad , p\epsilon P,\ q\epsilon Q,$$

satisfait les contraintes doubles

$$\sum_{q \in Q} T_{pq} = O_p \qquad , p \in P,$$

$$\sum_{p \in P} T_{pq} = D_q \qquad , q \in Q,$$

où O_p est la production connue de la zone p et D_q est l'attraction connue de la zone q. Le problème est résolu par un algorithme itératif, en ajustant successivement les α_p et les β_q (Furness 1970). Le problème de balancement à trois dimensions est une légère généralisation du problème ci-haut, proposée initialement par Evans et Kirby (1974). Dans ce cas, les paires origine - destination sont partitionnées en K catégories. En plus des contraintes de production et d'attraction données ci-haut, on impose aussi le nombre total de voyages par catégorie k. Soit $\{x_{pq}\}$, la matrice des indices de catégories, i.e. la paire (pq) appartient à la catégorie x_{pq}. Le troisième jeu de contraintes devient alors

$$\sum_{x_{pq}=k} T_{pq} = R_k \qquad , k \in K,$$

La matrice résultante est de la forme

$$T_{pq} = \alpha_p \beta_q \gamma_{x_{pq}} t_{pq} \qquad , p \in P, q \in Q.$$

L'implantation de ce modèle demande la spécification d'une matrice origine contenant les productions O_p, d'une matrice destination contenant les attractions D_q, et de deux matrices pleines: $\{t_{pq}\}$, la matrice à balancer et $\{x_{pq}\}$, la matrice des indices de catégories. Puisque typiquement, il n'y a que peu de contraintes de la troisième dimension, les valeurs de R_k sont entrées interactivement. Parmi les matrices résultantes, $\{\alpha_p\}$, $\{\beta_q\}$ et $\{T_{pq}\}$, l'usager choisit lesquelles il désire stocker dans la banque de

données. Les catégories k peuvent être utilisées pour tenir compte de toutes sortes d'informations additionnelles disponibles, comme la distribution de longueur de voyages, des comptages-coupes (anglais "screen line counts"), etc. Normalement, la calculatrice de matrice (module 3.21) est utilisée pour générer la matrice $\{x_{pq}\}$. La plupart des modèles de distribution, se réduisent à des problèmes de balancement, une fois que la matrice de coûts interzonaux, i.e. la matrice à balancer, est évaluée. Présentement, ce module n'incorpore pas encore le modèle de balancement Poissonien, qui a été développé dans le chapitre II, mais son ajout est prévu sous peu.

5.4 L'éditeur de fonctions

Dans ce groupe de modules, on trouve tous les outils nécessaires pour entrer, modifier, afficher et tabuler des fonctions. Les modules traitent les fonctions d'une façon unifiée, sans égard aux différentes classes de fonctions. Les modules de ce groupe constituent en soi un ensemble d'outils autonomes, car ils ne font appel à aucun autre type de données de EMME/2. Ils pourraient donc très bien être utilisés indépendamment du contexte EMME/2, comme un système de traitement de fonctions, ou ils pourraient être intégrés tels quels dans d'autres systèmes interactifs qui utilisent des relations fonctionnelles définies par l'usager.

Le module 4.11, "Entrée séquentielle de fonctions", lit un fichier séquentiel qui contient la définition d'une ou de plusieurs fonctions. Une fonction est spécifiée par son identificateur, un signe d'égalité "=" et une expression algébrique, tel que décrite dans la section 4.3.

Le module 4.12, "Entrée / modification interactive de fonctions", permet à l'usager d'ajouter ou d'éliminer des fonctions interactivement. De plus, ce module affiche les mots-clefs des variables associées à chacune des classes, ainsi que les opérateurs et les fonctions intrinsèques disponibles.

Le module 4.13, "Affichage graphique de fonctions", génère des dessins contenant une ou plusieurs courbes de fonctions de la même classe. Les valeurs des fonctions sont affichées sur l'axe vertical; l'axe horizontal contient une des variables associées à la classe de fonction. Les valeurs des autres variables sont spécifiées par l'usager pour chaque courbe et restent fixes pour l'affichage du dessin. Ce module est illustré par deux exemples. Dans le premier exemple, voir figure III-5-12, on retrouve des fonctions volume-délai, reliant le temps de parcours d'un lien avec le degré d'utilisation de ce lien. Le temps (en minutes) requis pour franchir la distance d'un mille est affiché en fonction du nombre de véhicules par heure et voie. Les cinq courbes correspondent à cinq catégories de routes différentes. Le deuxième exemple, montré dans la figure III-5-13, illustre l'utilisation de la classe des fonctions de l'usager pour l'analyse de fonctions quelconques, qui ne sont pas nécessairement reliées à la planification des transports. L'exemple montre les 8 premières approximations de Fourier d'une fonction de rampe triangulaire.

Le module 4.14, "Sortie alphanumérique et tabulation de fonctions", permet à l'usager de générer des rapports et des tables séquentielles contenant les descriptions des fonctions. De plus, l'usager se sert de ce module pour générer des rapports qui contiennent des valeurs tabulées d'une ou de plusieurs fonctions.

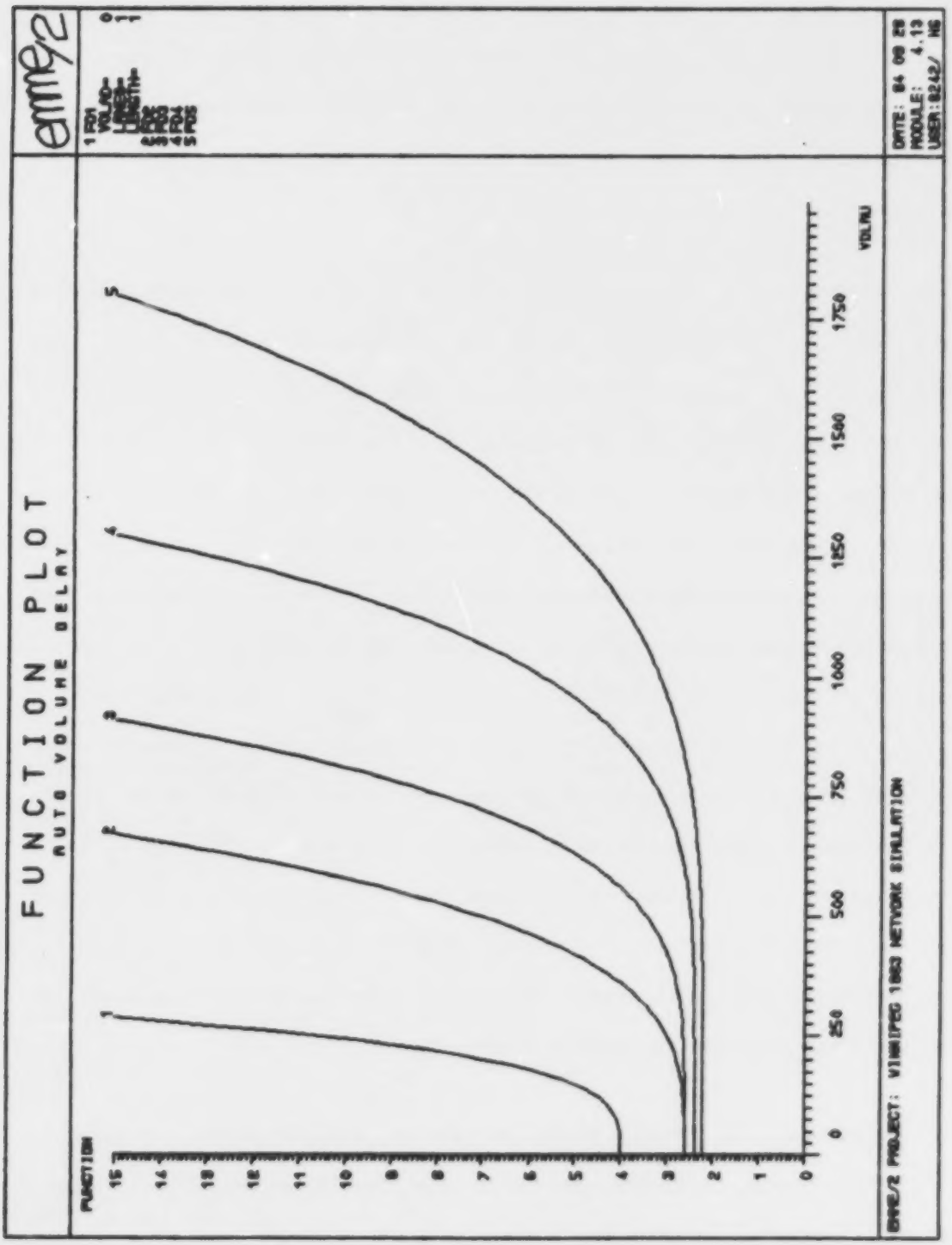

FIGURE III-5-12: Fonctions volume - délai.

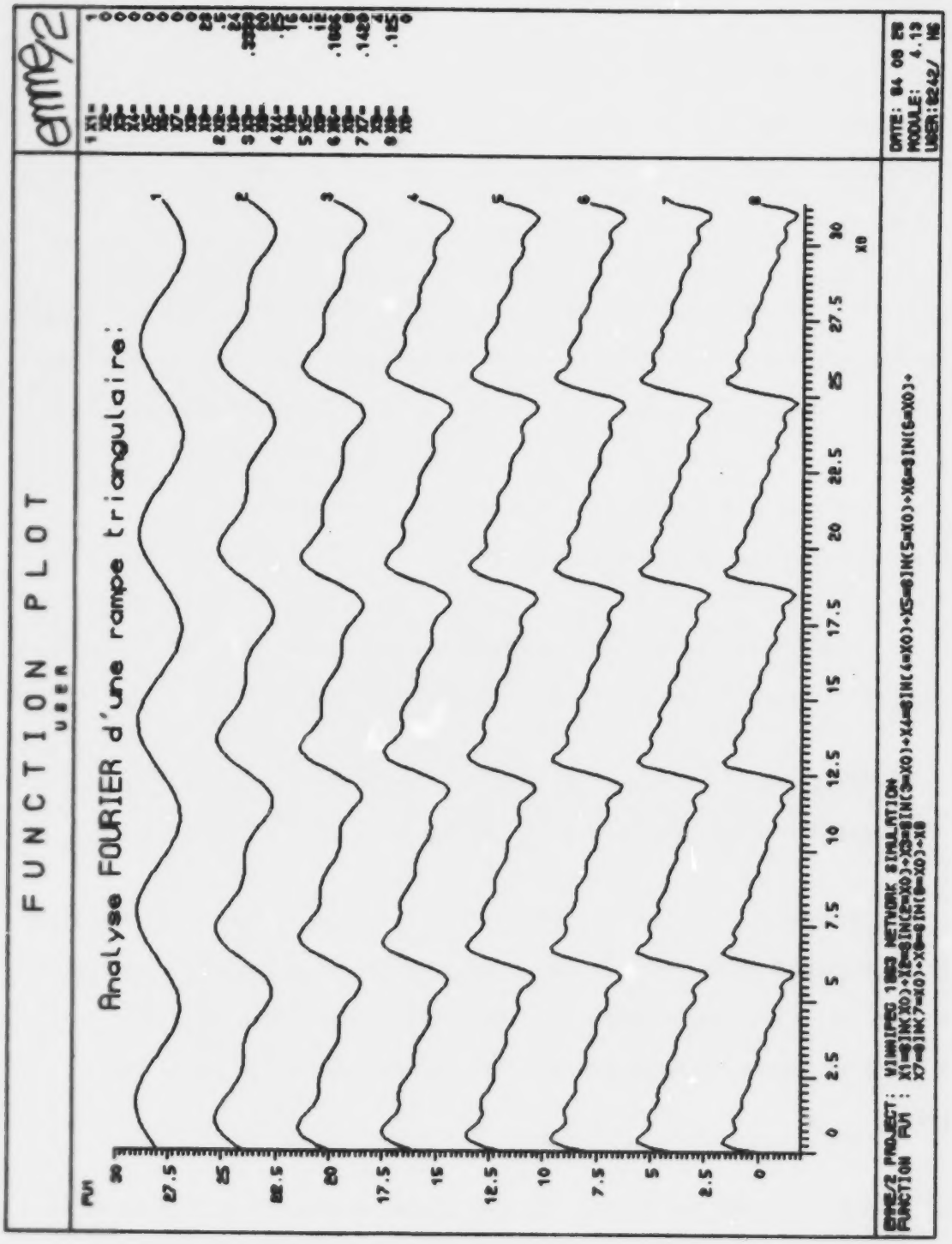

FIGURE III-5-13: L'analyse graphique d'une fonction de l'usager.

5.5 Les procédures d'affectation

Dans ce groupe de modules, on trouve un module interactif qui permet à l'usager de définir les paramètres de l'affectation désirée, ainsi que des modules non-interactifs, contenant les procédures d'affectation proprement dites.

Le modèle d'affectation le plus général prévu dans EMME/2, est un modèle d'affectation d'équilibre multimodal avec des fonctions de demande directe. Puisque ce modèle sera implanté d'une façon très modulaire, l'usager aura accès à un grand choix de sous-modèles de ce modèle complexe, pour l'adapter à ses besoins. A présent (septembre 1984), deux types d'affectation sont disponibles dans EMME/2:

- Affectation d'équilibre du réseau routier à demande fixe avec pénalités de virages sur les noeuds d'intersection.

- Affectation du transport en commun à demande fixe selon le modèle développé dans le chapitre II, section 1.

Evidemment, il est possible d'utiliser le modèle d'affectation d'équilibre du réseau routier comme une affectation de type "tout-ou-rien", soit en utilisant des fonctions volume - délai constantes, soit en limitant à zéro le nombre d'itérations de l'algorithme de Frank et Wolfe (voir plus loin).

Dans les pages suivantes, nous décrirons les modules existants et nous donnerons des détails d'implantation des deux modules d'affectation.

Le module 5.11, "Préparation d'un scénario pour une affectation", permet à l'usager de définir exactement le type d'affectation désiré, ainsi que tous les paramètres et ensembles de données pertinents pour effectuer cette affectation. De plus, ce module effectue certains tests préliminaires sur les ensembles de données spécifiés pour éviter plus tard des conditions d'erreurs dans les modules d'affectation.

Pour une affectation routière à demande fixe sur le scénario courant, l'usager doit spécifier les paramètres et les ensembles de données suivants:

- matrice de demande (personnes)
- matrice de taux d'occupation d'auto (personnes/véhicule)
- matrice de demande additionnelle (véhicules)
- volumes additionnels (aucun, véhicules t.c., données de l'usager ou les deux)
- matrice pour stocker les impédances O-D
- critères d'arrêt (nombre maximal d'itérations, pourcentage admissible du "gap", différence négligeable entre temps de voyage moyen et temps de voyage minimal)

Pour une affectation de transport en commun à demande fixe, l'usager doit spécifier les paramètres et ensembles de données suivants:

- matrice de demande (personnes)
- temps de transfert (temps constant ou temps spécifiques par noeud, en utilisant un item de données de l'usager)
- facteur de temps d'attente qui relie l'intervalle combiné au temps d'attente des voyageurs (facteur constant ou facteurs spécifiques par noeud, en utilisant un item de données de

l'usager)

- facteurs de perception des diverses composantes du voyage (attente, transfert, déplacement par mode de transport en commun auxiliaire)
- matrices pour stocker les impédances O-D et, si désiré, leurs diverses composantes

Le module 5.21, "Affectation routière", contient les procédures nécessaires pour résoudre le modèle d'affection d'équilibre à demande fixe suivant:

$$\text{Min} \sum_{a\in A} \int_0^{v_a} s_a(v+f_a)dv + \sum_{i\in I} \sum_{a_1\in A_i^-} \sum_{a_2\in A_i^+} \int_0^{v_{a_1a_2}} p_{a_1a_2}(v+f_{a_1a_2})dv$$

sujet à

$$v_a = \sum_{k\in K} \delta_{ak} h_k \qquad , a\in A,$$

$$v_{a_1a_2} = \sum_{k\in K} \delta_{a_1k} \delta_{a_2k} h_k \qquad , a_1\in A_i^-,\ a_2\in A_i^+,\ i\in I,$$

$$\sum_{k\in K_{pq}} h_k = \frac{q_{pq}}{\eta_{pq}} + \gamma_{pq} \qquad , p\in P,\ q\in Q,$$

$$h_k \geq 0 \qquad , k\in K_{pq},\ p\in P,\ q\in Q,$$

où on fait appel à la notation suivante:

indices et ensembles:

$p \in P$	zones origines
$q \in Q$	zones destinations
$i \in I$	noeuds du réseau routier
$a \in A$	liens du réseau routier
$a \in A_i^-$	liens entrant au noeud i
$a \in A_i^+$	liens sortant du noeud i
$k \in K_{pq}$	chemins reliant p et q
$k \in K$	tous les chemins
$i \in \bar{I}$	noeuds intersections (avec virages pénalisés)

constantes:

δ_{ak}	1 si lien a appartient au chemin k, 0 autrement
g_{pq}	demande auto de p à q en personnes
η_{pq}	taux d'occupation d'auto (personnes/véhicule)
γ_{pq}	demande additionnelle en véhicules
ξ_a	volume additionnel sur lien a en véhicules
$\xi_{a_1 a_2}$	volume additionnel sur le virage $(a_1 a_2)$

relations fonctionnelles:

$t_a(v)$	fonction volume-délai sur lien a
$p_{a_1 a_2}(v)$	fonction de pénalité sur le virage $(a_1 a_2)$

variables:

v_a	volume prédit sur le lien a
$v_{a_1 a_2}$	volume prédit sur le virage $(a_1 a_2)$
h_k	volume sur le chemin k

Ce modèle est résolu par la méthode des approximations linéaires successives (Frank et Wolfe, 1956; Nguyen, 1974; Leblanc et al., 1975). L'avantage principal de cette approche est qu'elle évite de stocker explicitement les volumes sur les chemins h_k, $k \in K$, en réduisant ainsi, radicalement, le nombre de variables à conserver. L'idée de base de cette méthode est de trouver, à partir de la solution réalisable courante, $\hat{v}$, une direction de descente $(\tilde{v}-\hat{v})$ en linéarisant la fonction objective, i.e.

$$\text{Min} \sum_{a \in A} \tilde{v}_a s_a(\hat{v}_a) + \sum_{i \in \bar{I}} \sum_{a_1 \in A_i^-} \sum_{a_2 \in A_i^+} \tilde{v}_{a_1 a_2} p_{a_1 a_2}(\hat{v}_{a_1 a_2})$$

et de résoudre le problème résultant pour les variables $\tilde{v}$. Ce problème linéaire se trouve être un problème de chemins plus courts avec des pénalités de virages. Il s'agit donc d'un problème bien connu (Potts et Oliver, 1972) qui peut être résolu, soit en "explosant" le réseau (chaque lien est transformé en sommet, chaque virage est transformé en arc), soit en adaptant l'algorithme de plus courts chemins (on donne une étiquette par lien, au lieu d'une étiquette par noeuds, comme c'est le cas dans le problème de plus courts chemins ordinaire). Ce qui rend le problème ici plus intéressant est le fait que les virages pénalisés ne se trouvent que sur un sous-ensemble des noeuds, soit $\bar{I}$. Aux noeuds réguliers, tous les virages sont permis sans pénalité, sauf les virages en U, qui sont interdits par définition. La proposition suivante nous permettra d'exploiter ce fait dans un algorithme efficace.

Proposition: Soit $(i_0, i_1, i_2, \ldots, i_{N-1}, i_N)$ la séquence de noeuds décrivant un plus court chemin k reliant i_0 et i_N sur un réseau avec coûts positifs sur les liens et pénalités de virage non-négatives sur un sous-ensemble de noeuds $\bar{I}$. Si le même noeud i apparaît deux fois dans le chemin

k, i.e. $i=i_m=i_n$, où $m < n$, alors il existe un noeud $i_{m'} \in \bar{I}$ avec $m' \leq m$ et un noeud $i_{n'} \in \bar{I}$ avec $n' \geq m$.

Preuves: Supposons que la proposition ne tient pas, i.e. il n'existe pas de tels noeuds $i_{m'}$ et $i_{n'}$. Ceci implique donc que le chemin $k' = (i_0, i_1, \ldots, i_{m-1}, i_m=i_n, i_{n+1}, \ldots, i_N)$ est un chemin qui ne contient aucun virage pénalisé, car il ne contient aucun noeud qui appartient à $\bar{I}$. La longueur du chemin k est alors la somme de la longueur du chemin k' et la longueur du circuit $(i_m, i_{m+1}, \ldots, i_{n-1}, i_n=i_m)$. Puisque ce circuit contient au moins un lien, sa longueur est strictement positive. Ceci implique que le chemin k' est un chemin de longueur plus petite que celle du chemin k - ce qui est une contradiction au fait que k est un plus court chemin.■

La proposition ci-haut nous apprend qu'on n'aura plus à considérer tous les liens $a=(i,j)$, où le chemin le plus court de la racine p au noeud j est déja connu et ne contient pas de noeud d'intersection $i' \in \bar{I}$. Ce résultat nous permet de diminuer considérablement l'effort requis pour calculer les plus courts chemins d'une origine p à toutes les destinations $q \in Q$. Rappelons que, en général, l'algorithme de plus courts chemins avec pénalités de virage calcule les chemins d'un noeud (ou lien) à tous les liens du réseau. La modification proposée permet d'éliminer systématiquement les chemins vers des liens qui ne peuvent pas faire partie d'un plus court chemin d'une origine à une destination. Pour le cas spécial $\bar{I}=\emptyset$, il résulte de cette modification une économie de environ 50%, i.e. cet algorithme n'est que légèrement plus lent qu'un algorithme avec étiquettes sur les noeuds.

L'algorithme suivant illustre l'implantation de la procédure de calcul de plus courts chemins utilisée dans les modules 5.21 et 6.15 (voir section suivante). A partir d'un réseau avec coûts c_a sur les liens et $c_{a_1 a_2}$ sur les virages aux noeuds d'intersections $\bar{I}$, l'algorithme calcule les plus courts chemins d'une origine p à toutes les destinations $q \in Q$, ainsi que leur longueur u_q. Les chemins sont stockés à l'aide de pointeurs vers le lien prédécesseur (soit b_q le dernier lien du chemin à la destination q et b_a le lien prédecesseur du lien a). Le chemin de p à q peut être parcouru en sens inverse; il est de forme $(a_1=b_{a_2}, a_2=b_{a_3}, \ldots, a_N=b_q)$, où $b_{a_1}=0$. La distance de l'origine p au lien a est stockée dans u_a. De plus, l'algorithme utilise une variable booléenne, Ψ_i, pour chaque noeud $i \in I$. Ψ_i a une valeur de 1, si le chemin de p à i (inclusivement) ne contient pas d'intersection, 0 autrement. L'ensemble de liens $\bar{A}$ dénote les liens qui sont atteints, mais doivent encore recevoir une étiquette permanente.

Calcul des plus courts chemins avec virages pénalisés:

ETAPE 0: (Initialisation)

$u_q := \infty$, $b_q := \infty$, $q \in Q$;

$\Psi_i := 0$, $i \in I-(p)$; $\Psi_p := 1$;

$u_a := \infty$, $b_a := \infty$, $a \in A - A_p^+$;

$u_a := c_a$, $b_a := 0$, $a \in A_p^+$;

$\bar{A} := A_p^+$;

ETAPE 1: (Sélection des liens à balayer)

Tant que $\bar{A} \neq \emptyset$ faire:

a) déterminer $\bar{a}=(i,j) \in \bar{A}$ tel que $u_{\bar{a}} \leq u_a$, $a \in \bar{A}$

b) si $j \in \bar{Q}$ effectuer ETAPE 2, (destination)

si $j \in \bar{I}$ effectuer ETAPE 3, (intersection)

autrement effectuer ETAPE 4; (noeud régulier)

c) $\bar{A} := \bar{A} - \{\bar{a}\}$.

Ensuite ARRETER.

ETAPE 2: (Noeud j est une destination)

$u_q := u_{\bar{a}}$; $b_q := \bar{a}$.

ETAPE 3: (Noeud j est une intersection)

Pour $a=(j,l) \in A_j^+$ tels que $\varphi_l = 0$ faire:

si $u_{\bar{a}} + c_{\bar{a}a} + c_a < u_a$ alors $u_a := u_{\bar{a}} + c_{\bar{a}a} + c_a$; $b_a := \bar{a}$; $\bar{A} := \bar{A} + \{a\}$.

ETAPE 4: (Noeud j est un noeud régulier)

Pour $a=(j,l) \in A_j^+$ tels que $l \neq i$ faire:

si $u_{\bar{a}} + c_a < u_a$ alors $u_a := u_{\bar{a}} + c_a$; $b_a := \bar{a}$; $\bar{A} := \bar{A} + \{a\}$; $\varphi_j := \varphi_i$.

Evidemment, cet algorithme ne donne que l'idée générale de l'implantation, sans entrer dans les détails d'optimisations additionnelles possibles. Voici quelques points portant sur une implantation efficace de l'algorithme ci-haut:

- L'ensemble $\bar{A}$ est organisé en structure de tas, pour ainsi rendre plus efficace la recherche du lien $\bar{a}$.

- On remarque qu'on utilise les étiquettes de liens u_a seulement pour les liens actÃ. On peut donc sauver de l'espace en gardant cette information dans une structure qui est parallèle à la structure de tas.

- L'étape 4 peut être optimisée en ne faisant le test sur les étiquettes que pour les premiers liens, car tous les tests subséquents ont nécessairement le même résultat. Ceci implique de plus que Ψ_i ne doit être mis à jour qu'au plus une fois.

- Dans l'étape 2, on s'assure, d'une façon élégante, de ne jamais considérer des chemins qui traversent des centroïdes.

Il est à noter que l'algorithme donné ci-haut, est un algorithme exact, i.e. il trouve toujours le chemin le plus court dans tous les cas où il existe au moins un chemin. Cette approche se distingue radicalement des approches heuristiques qui sont utilisées dans beaucoup de logiciels d'affectation, comme par exemple TRAFFIC, ICAM et EMME/1. Ces heuristiques consistent en une "explosion", manuelle ou automatique, des noeuds d'intersection telle que chaque virage de l'intersection correspond à un lien auxiliaire. Un algorithme de plus court chemin ordinaire, i.e. sans virages pénalisés, est ensuite appliqué à ce réseau "explosé". Ces heuristiques sont sujettes à certaines aberrations: soit qu'elles admettent des chemins comportant des virages en U aux noeuds réguliers, soit qu'elles produisent des chemins qui ne sont pas optimaux ou, dans le pire cas, ne trouvent même pas de chemin du tout pour certaines paires O-D.

Une fois qu'on sait calculer les plus courts chemins, on n'a qu'à

appliquer l'algorithme au réseau routier, en utilisant les coûts

$$c_a = s_a(\hat{v}_a) \qquad , a \in A,$$

$$c_{a_1a_2} = p_{a_1a_2}(\hat{v}_{a_1a_2}) \qquad , a_1 \in A_i^-,\ a_2 \in A_i^+,\ i \in \bar{I},$$

et à affecter la demande g_{pq} à ces chemins, pour obtenir les volumes $\tilde{v}$. La longueur du pas optimal dans la direction de descente $(\tilde{v}-\hat{v})$, λ^*, est celle qui minimise la fonction objective, soit

$$\text{Min } f(\lambda) = \sum_{a \in A} \int_0^{(1-\lambda)\hat{v}_a+\lambda\tilde{v}_a} s_a(v)dv +$$

$$+ \sum_{i \in \bar{I}} \sum_{a_1 \in A_i^-} \sum_{a_2 \in A_i^+} \int_0^{(1-\lambda)\hat{v}_{a_1a_2}+\lambda\tilde{v}_{a_1a_2}} p_{a_1a_2}(v)dv$$

Comme on le verra, il est préférable de ne pas effectuer cette minimisation telle quelle, mais plutôt d'analyser le gradient de la fonction $f(\lambda)$

$$\frac{df(\lambda)}{d\lambda} = \sum_{a \in A} s_a((1-\lambda)\hat{v}_a+\lambda\tilde{v}_a)(\tilde{v}_a-\hat{v}_a) +$$

$$+ \sum_{i \in \bar{I}} \sum_{a_1 \in A_i^-} \sum_{a_2 \in A_i^+} p_{a_1a_2}((1-\lambda)\hat{v}_{a_1a_2}+\lambda\tilde{v}_{a_1a_2})(\tilde{v}_{a_1a_2}-\hat{v}_{a_1a_2})$$

Si $\frac{df(\lambda=0)}{d\lambda} \leq \varepsilon$, alors on arrête avec la solution $v^*=\hat{v}$, puisqu'on a trouvé les volumes d'équilibre. Si $\frac{df(\lambda=1)}{d\lambda} < 0$, alors on met $\lambda^* = 1$, i.e. on remplace $\hat{v}$ par $\tilde{v}$. Autrement, la valeur optimale de λ est celle qui annulle le gradient, soit $\frac{df(\lambda)}{d\lambda} = 0$, $0 \leq \lambda \leq 1$.

Le fait de travailler avec le gradient, au lieu de minimiser la fonction telle quelle, a plusieurs avantages importants:

- Les algorithmes pour chercher le zéro d'une fonction monotone sont, en général, plus efficaces que ceux pour minimiser une fonction convexe. Mentionnons ici seulement l'algorithme de bisection, qui à chaque itération réduit l'intervalle de recherche par un facteur de 2, et comparons-le avec l'algorithme de la section dorée qui demande plus de travail par itération et ne réduit l'intervalle de recherche que d'un facteur 1.618 à chaque itération.

- La recherche d'un zéro d'une fonction est numériquement plus stable que la recherche d'une valeur extrême. Ceci est une conséquence du fait que la précision des nombres reels, stockés en format de point flottant, augmente en s'approchant de zéro. Dans une minimisation , par contre, plus on s'approche du minimum, plus on a affaire avec des différences de nombres très similaires ce qui, fait bien connu, cause des problèmes d'instabilité numérique. Dans les problèmes d'affectation, ce problème est particulièrement grave, puisque la fonction objective contient une très grande composante constante.

- L'évaluation du gradient est plus simple que l'évaluation de la fonction objective. Dans notre cas, le gradient ne contient que des sommes, des multiplications et des appels aux fonctions volume-délai et pénalité de virage. On n'y trouve plus les intégrales qui rendent compliquée l'évaluation de la fonction objective. Ceci est encore plus important étant donné que EMME/2 traite les fonctions comme des éléments de la banque de données (voir section 3.3) et, par

conséquent, n'a pas accès aux formes analytiques des intégrales.

Dans EMME/2, les fonctions volume-délai et les fonctions de pénalité de virage dépendent de plusieurs variables en plus du volume prédit (distance, nombre de voies, volume additionnel, données de l'usager, etc.). Il va de soi qu'il est impossible d'amener toutes ces données, pour tout le réseau, en même temps en mémoire centrale. L'effort requis pour évaluer le gradient pour une valeur donnée de λ ne consiste donc pas seulement dans l'évaluation des fonctions, mais, facteur plus important, il faut aussi considérer le temps d'accès requis pour lire les variables nécessaires dans la banque de données. C'est pour cette raison que l'utilisation d'algorithmes calculant les valeurs de λ, itération par itération, n'est pas efficace dans notre contexte. Il faut plutôt utiliser une méthode qui détermine d'avance une série de valeurs de λ et calcule ensuite en parallèle les valeurs correspondantes du gradient. La valeur optimale λ^* peut, par la suite, être trouvée par interpolation. Dans la procédure implantée dans le module 5.21, nous avons choisi la forme $\lambda_i = \frac{\lambda^{max}}{\alpha^i}$, $i=1,\ldots,i_{max}$. pour ces valeurs d'échantillonnage du gradient. Une interpolation quadratique est appliquée aux trois plus petites valeurs du gradient, pour estimer la longueur du pas optimal λ^*. L'expérience a montré que le choix des paramètres $\alpha=2$ et $i_{max}=6$ semble adéquat pour toutes les applications typiques.

Portons maintenant notre attention aux procédures d'affectation du transport en commun. Le module 5.3i, "Affectation du transport en commun", contient une implantation du modèle de choix de route optimale qu'on a développé dans la section 1 du chapitre II. Il n'est donc pas nécessaire de répéter ici la formulation du modèle. Nous nous concentrerons plutôt sur la représentation du réseau et sur l'adaptation de l'algorithme à cette représentation.

Pour le développement et l'analyse des propriétés mathématiques du modèle, nous avons supposé que le réseau de transport en commun avait été transformé préalablement en un réseau généralisé. Ce réseau transformé fait abstraction des lignes de transport en commun; il ne connaît que des noeuds et des liens (qui ne correspondent pas nécessairement aux noeuds et aux liens du réseau physique). Chaque lien comporte un coût et une fréquence. Pour un lien à service continu, i.e. sans attente, la fréquence est mise à l'infini. Cette méthode de représentation d'un réseau de transport en commun permet de formuler le problème d'une façon très générale et elle est idéale pour analyser mathématiquement le modèle. Pour le but d'une implantation efficace du modèle, par contre, cette représentation a plusieurs désavantages sérieux:

- La transformation du réseau de transport en commun du format EMME/2 en ce type de réseau généralisé, et vice versa, augmente considérablement le temps d'exécution.

- Le réseau généralisé occupe beaucoup plus d'espace mémoire que le réseau de transport en commun en format EMME/2.

- Le réseau généralisé contient beaucoup d'informations redondantes; on n'a qu'à penser aux fréquences qui, pour tous les arrêts de la même ligne, sont identiques et qui sont infinies pour tous les liens autres que ceux d'embarquement.

- La structure du réseau généralisé est hautement spécialisée. L'algorithme, dans sa formulation générale, ne peut pas tirer avantage de ce fait.

Pour ces raisons, nous avons décidé de ne pas adapter le réseau à l'algorithme mais, au contraire, d'adapter l'algorithme à la structure du réseau de transport de EMME/2.

Dans la section 3.2 nous avons vu la structure interne du réseau de transport en commun, i.e. des lignes de transport en commun et des liens de mode de transport en commun auxiliaires. Avant d'énoncer l'algorithme, tel qu'il est implanté dans le module 5.31, il nous faut définir une notation mathématique adéquate:

Indices et ensembles:

$i \in I$	noeuds du réseau routier
$a \in A$	liens des modes de transport en commun auxiliaire
$a \in A_i^-$	liens entrant au noeud i
$a \in A_i^+$	liens sortant du noeud i
$l \in L$	lignes de transport en commun
$s \in S_l$	arrêts de la ligne l
$s \in S^i$	arrêts au noeud i

constantes:

t_s	temps de déplacement sur le segment suivant l'arrêt s
t_a	temps de déplacement sur le lien a
p_i	pénalité d'embarquement au noeud i
α_i	facteur de temps d'attente au noeud i
f_l	fréquence de la ligne l
g_i	demande, i.e. nb de voyages du noeud i à la destination q

variables:

u_s	impédance de l'arrêt s à la destination q (sans attente)
u_i	impédance du noeud i à la destination q en quittant le noeud par une ligne de transport en commun
u_i'	impédance du noeud i à la destination q en quittant le noeud par un lien de t.c. auxiliaire
b_s	élément successeur de l'arrêt s
b_i	élément successeur du noeud i
f_i	fréquence combinée au noeud i
S^{gi}	arrêts attrayants au noeud i
v_s	volume sur le segment suivant l'arrêt s (cumulatif)
v_a	volume sur le lien a (cumulatif)
v_i	volume temporaire sur le noeud i

En utilisant cette notation, l'algorithme suivant illustre l'implantation du modèle d'affectation des voyages du transport en commun, comme elle est réalisée dans le module 5.31. Evidemment, l'algorithme donné plus bas ne donne qu'un aperçu général de l'approche, sans entrer dans les détails de programmation.

PARTIE 1: Calcul des stratégies optimales

ETAPE 0: (Initialisation)

$u_s := \infty$, $s \in S_1$, $l \in L$;

$u_i := \infty$, $u_i' := \infty$, $i \in I-(q)$; $u_q := 0$;

$f_i := 0$, $v_i := g_i$, $i \in I$;

$\tilde{I} := (q)$, $\tilde{S} := \emptyset$, $\bar{S} = \emptyset$.

ETAPE 1: (Sélection des éléments)

Tant que $\tilde{I} \cup \tilde{S} \neq \emptyset$ faire:

a) déterminer $j \in \tilde{I}$ tel que $\text{Min}(u_j, u_j') \leq \text{Min}(u_i, u_i')$, $i \in \tilde{I}$;

b) déterminer $s \in \tilde{S}$ tel que $u_s \leq u_{s'}$, $s' \in \tilde{S}$;

c) si $\tilde{S} = \emptyset$ ou $\text{Min}(u_j, u_j') < u_s$ alors effectuer ETAPE 2,

autrement effectuer ETAPE 3.

ETAPE 2: (Balayage du noeud j)

a) si $u_j' < u_j$ alors $u_j := u_j'$, $\bar{S}^j = \emptyset$;

b) pour chaque arrêt $s \in S^j$ faire:

si $u_s > u_j$ alors $u_s := u_j$, $b_s := j$, $\tilde{S} := \tilde{S} \cup (s)$;

c) pour chaque lien $a=(i,j) \in A_j^-$ faire:

si $u_i' > u_j$ alors $u_i' := u_j + t_a$, $b_i := j$, $\tilde{I} := \tilde{I} \cup (i)$;

d) $\tilde{I} := \tilde{I} - (j)$.

ETAPE 3: (Balayage de l'arrêt s)

Soit i et l tels que $s \in S^i$ et $s \in S_l$, et soit s' l'arrêt précédent sur l'itinéraire de la ligne l, s'il existe.

a) Si s' existe et $u_{s'} > u_s + t_{s'}$, alors $u_{s'} := u_s + t_{s'}$, $b_{s'} := s$, $\tilde{S} := \tilde{S} \cup (s')$;

b) si $u_i > u_s + p_i$ alors

$$S^i := S^i + (s),\quad u_i := \frac{f_i u_i + f_l (u_s + p_i)}{f_i + f_l};\quad f_i := f_i + f_l,\quad \tilde{I} := \tilde{I} \cup (i); \quad (*)$$

c) $\tilde{S} := \tilde{S} - (s)$.

PARTIE 2: Affectation des voyages au réseau

ETAPE 4: (Balayage du réseau en ordre topologique inversé des noeuds)

Pour tout $i \in I - (q)$, en ordre décroissant de u_i, faire:

si $S^i = \emptyset$ alors effectuer ETAPE 5,

autrement effectuer ETAPE 6.

ETAPE 5: (Chargement d'un lien)

$a = (i,j) := b_i$, $v_a := v_a + v_i$, $v_j := v_j + v_i$.

ETAPE 6: (Chargement des segments de lignes)

Pour chaque arrêt $s \in S^i$ faire:

a) $\nu := \frac{f_l}{f_i} \cdot v_i$, $v_s := v_s + \nu$, $e := s$;

b) répéter: $e := b_e$, $v_e := v_e + \nu$ jusqu'à $e \in I$.

(*) Lorsqu'on ajoute le premier arrêt attrayant à S^i, on a que $u_s = \infty$ et $f_i = 0$. Nous suivons la même convention que dans le chapitre II, i.e. nous mettons $0 \cdot \infty = \infty$.

L'algorithme ci-haut est assez différent de l'algorithme original, bien que le modèle de base reste inchangé. Voici quelques points importants pour clarifier le fonctionnement, ainsi que des suggestions pour une implantation efficace de l'algorithme:

- Les ensembles $\tilde{S}$ et $\tilde{I}$ contiennent les arrêts et les noeuds qui sont déjà atteints, mais qui n'ont pas encore été examinés. Ces deux ensembles peuvent être réalisés à l'aide d'une seule structure de tas qui contient en même temps les deux types d'élément. Ceci permet de réunir les étapes 1a) et 1b) et simplifie ainsi la sélection du prochain élément à examiner.

- Selon l'algorithme original, les arrêts devraient être examinés en ordre croissant de u_s+p_i. Dans cet algorithme, par contre, ils sont traités en ordre croissant de u_s. Il est facile de voir que, à un noeud donné, l'ordre reste le même, puisque la pénalité d'embarquement est la même pour tous les arrêts à ce noeud. Les liens auxiliaires sont examinés en ordre croissant de u_j (et non pas u_j+t_a, comme le demande l'algorithme original). Pour éviter des problèmes, il faut utiliser deux étiquettes distinctes par noeud, u_i et u_i'. Ces deux étiquettes peuvent être interprétées comme des étiquettes de deux "noeuds imaginaires": le premier (u_i) est connecté à tous les arrêts $s \in S^i$ et le deuxième (u_i') à tous les liens $a \in A_i^+$. Les deux "noeuds imaginaires" sont reliés par des "liens imaginaires", à coût nul, au "vrai" noeud i. L'étape 2a) peut alors être interprétée comme l'examen de ces "liens imaginaires".

- Les volumes sur les segments, v_s, et sur les liens, v_a, ne sont pas initialisés par l'algorithme. Ceci permet d'accumuler les volumes totaux pour plusieurs destinations, en appliquant l'algorithme plusieurs fois de suite.

- Parce que l'algorithme n'associe aucun coût à un débarquement, les arrêts qui sont améliorés par l'étape 2b) pourraient tout de suite être examinés.

- On note que l'élément successeur d'un arrêt s, b_s, ne peut être qu'une de deux possibilités: il s'agit soit de l'arrêt suivant sur l'itinéraire de la ligne, soit du noeud qui correspond à l'arrêt. On peut donc économiser de l'espace en stockant cette information dans un seul bit.

Dans notre présentation de l'algorithme, nous avons laissé de côté certains détails techniques de l'implantation. Voici certains aspects additionnels qui sont traités dans le module 3.31:

- nombre d'embarquements stocké pour chaque arrêt,
- lignes express qui n'arrêtent pas à chaque noeud,
- sections de lignes où les voyageurs ne peuvent qu'embarquer (ou débarquer),
- facteurs de pondération des diverses composantes du voyage pour tenir compte de la perception différente de chacune d'elles,
- traitement de l'arrêt prolongé aux terminus des lignes (en anglais "layover").

5.6 L'analyse des résultats

Les modules de ce groupe permettent à l'usager d'analyser les résultats d'une affectation. Par conséquent, ils ne sont appelés qu'après l'exécution d'une affectation sur le scénario courant. Les résultats principaux des affectations sont les temps de parcours et les volumes. Tous les résulats qui sont stockés sous forme de matrice ne sont pas traités par les modules de ce groupe. On utilise plutôt les outils de l'éditeur de matrice dont il a été question dans la section 5.3.

Pour le mode auto on distingue trois types de volumes. Premièrement, les **volumes auto** proprement dits, i.e. les volumes qui correspondent aux voyages faits par mode auto, affectés au réseau par un modèle de choix de route. Deuxièmement, les **volumes additionnels**; ils correspondent à des volumes fixes, qui influencent la congestion sur les liens, mais qui ne sont pas sujets à un choix de route par un modèle d'affectation. Dans EMME/2, les volumes additionnels proviennent d'un item de données de l'usager ou ils correspondent aux véhicules qui desservent les lignes de transport en commun (par l'intermédiaire du coefficient d'équivalence auto), ou une combinaison des deux. Finalement, les **volumes totaux** représentent les sommes des volumes auto et volumes additionnels.

Le module 6.11, "Rapports de temps et volumes - auto", génère des rapports concernant les résultats d'une affectation des voyages en automobile. Trois types de rapport sont disponibles:

- temps, vitesse et volumes sur les liens,
- temps et volumes sur les virages des carrefours,
- résumé de l'utilisation du réseau auto par type de lien.

Le module 6.12, "Affichage graphique de temps et volumes - auto", génère des dessins qui contiennent, pour chaque lien affiché, une des caractéristiques suivantes:

- volume auto
- volume additionnel
- volume total
- temps de parcours
- vitesse

Les valeurs peuvent être représentées graphiquement, par une bande à largeur proportionnelle parallèle au lien, numériquement ou par une combinaison des deux. L'exemple donné dans la figure III-5-14 montre les volumes totaux sur le réseau du mode auto au centre-ville.

Le module 6.13, "Comparaison graphique de temps et volumes - auto", compare graphiquement les résultats concernant le mode auto du scénario courant avec ceux d'un autre scénario. La comparaison est basée sur une des caractéristiques suivantes:

- volume auto
- volume additionnel
- volume total
- temps de parcours
- vitesse

Les différences, relatives ou absolues, des caractéristiques choisies pour les deux scénarios sont affichées, soit par des bandes proportionnelles, soit par des valeurs numériques, ou par les deux. Graphiquement, les différences positives et négatives sont distinguées par l'utilisation de différents styles de lignes (solides pour une différence positive, inter-

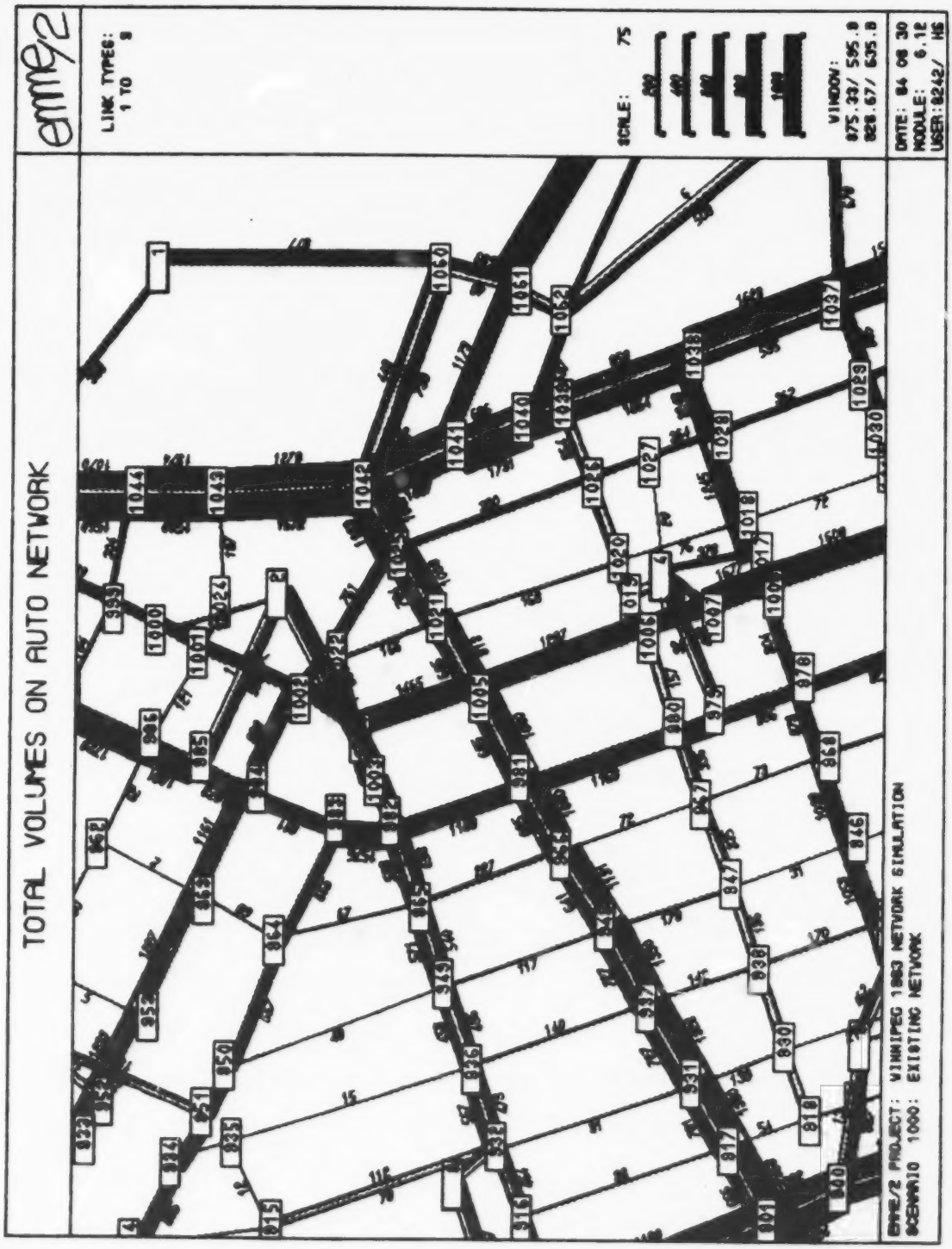

FIGURE III-5-14: Volumes - auto au centre-ville.

rompues pour une différence négative), ou, en présence d'un terminal à écran-couleur, par deux couleurs distinctes. Comme on l'a vu dans la section 3.1, chaque scénario contient un réseau complètement indépendent. Une comparaison de deux scénarios quelconques n'est donc, ni une tâche triviale à définir, ni facile à effectuer. Le plus grand problème est de trouver les éléments correspondants dans les deux scénarios: quel élement du premier scénario doit-on comparer à quel élément du deuxième scénario? Puisqu'il n'y a pas de correspondance interne, il faut définir une règle de correspondance qui fait appel aux attributs des éléments. Bien que, dans ce module, les comparaisons soient faites au niveau des liens, il suffit d'établir une règle de correspondance au niveau des noeuds; une fois cela fait, la correspondance des liens en découle. Pour les noeuds, deux attributs peuvent servir pour établir une telle correspondance: l'identificateur et les coordonnées. La correspondance par coordonnées essaie de coupler les noeuds selon leur proximité géographique. Il résulte de cette approche des procédures très coûteuses et des possibilités d'ambiguïtés de nomenclature. Avec la correspondance par identificateur, la méthode choisie dans EMME/2, on compare deux noeuds si et seulement si ils ont le même identificateur. Etant donné le stockage pré-trié du réseau, voir section 2.4 de ce chapitre, la correspondance des noeuds et des liens peut être trouvée par un simple algorithme de fusion (en anglais, "merge") qui a une complexite de calcul linéaire. Des ambiguïtés de location, i.e. deux noeuds correspondants qui ont des coordonnées différentes, sont résolues en donnant priorité au scénario courant. Si un lien n'existe que dans un des deux scénarios qu'on compare, la valeur zéro est utilisée pour l'autre scénario afin de pouvoir faire des différences. La figure III-5-15 montre un exemple d'une comparaison de volumes. Les deux scénarios se distinguent

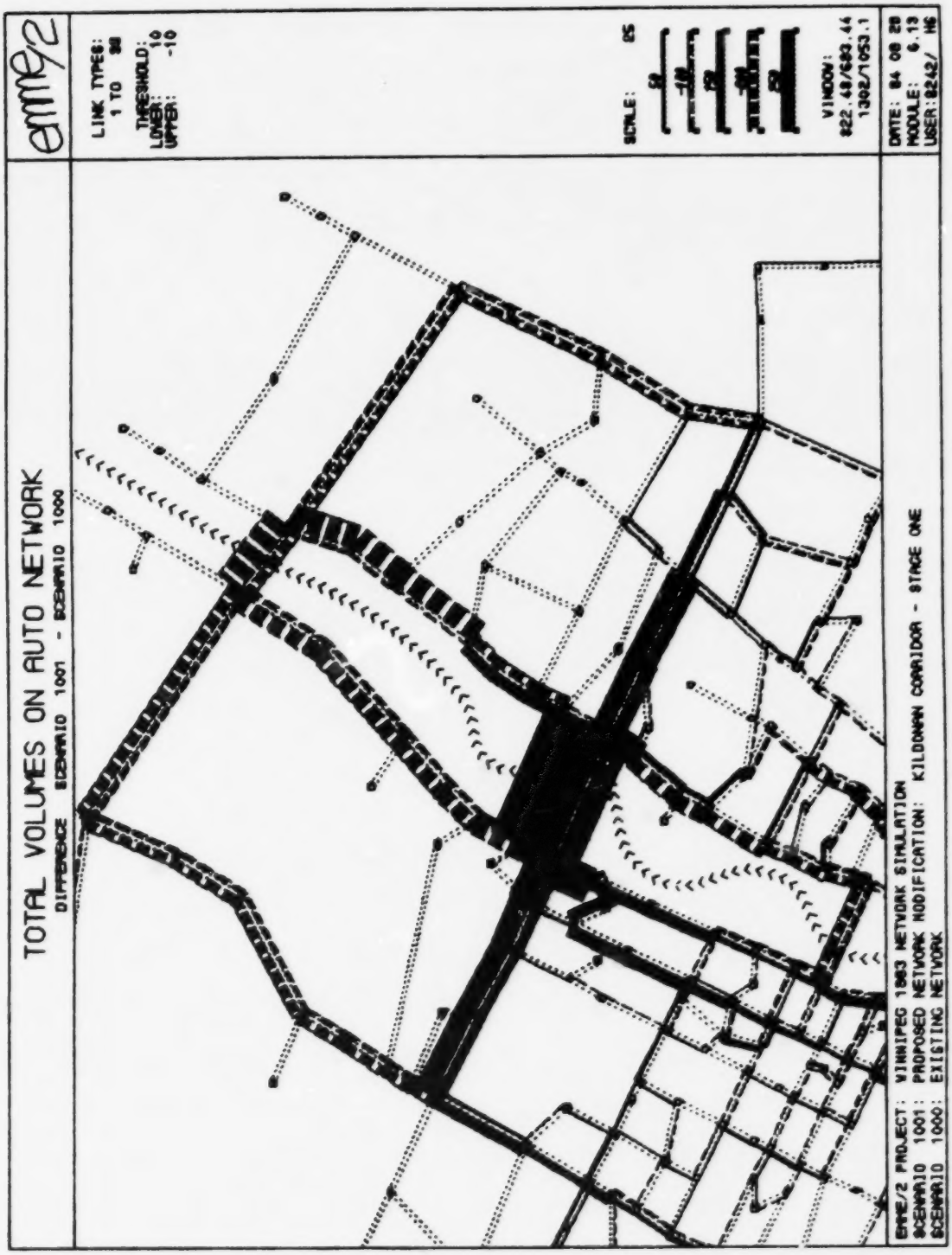

FIGURE III-5-15: Comparaison des volumes de deux scénarios.

par l'ajout d'un pont, situé au centre du dessin. Le dessin révèle les conséquences qu'aurait la construction du pont: soit les conséquences directes sur les voyages qui peuvent profiter d'un trajet plus direct à cause du pont, soit les conséquences indirectes sur des voyages qui n'utilisent pas le nouveau pont, mais qui vont changer leur trajet pour s'adapter aux nouvelles conditions de trafic.

Le module 6.14, "Affichage graphique des volumes sur les intersections", permet à l'usager de visualiser les flots sur les virages d'un ou de plusieurs carrefours. L'usager choisit le sous-ensemble de noeuds d'intersection qu'il désire afficher ainsi que le nombre d'intersections à afficher sur le même dessin. Chaque intersection est affichée avec les liens et noeuds adjacents; les volumes sur les virages sont représentés sous forme de bandes proportionnelles. En utilisant un terminal à écran-couleur, des couleurs différentes sont utilisées pour les virages provenant de différents liens. L'utilisation de ce module est illustrée dans la figure III-5-16.

Le module 6.15, "Accessibilité et plus courts chemins sur le réseau auto", sert à afficher des arbres de plus courts chemins sur le réseau du mode auto en tenant compte des virages pénalisés sur les intersections. Les chemins sont évalués soit en terme de temps de parcours, soit en terme de distance. Les dessins produits par ce module servent principalement à deux buts. Premièrement, ils permettent de visualiser les détails des chemins les plus courts et, par cela, le planificateur peut, par l'analyse de sous-arbres, tirer des conclusions sur l'utilité de certaines modifications du réseau. Toutefois, il ne faut pas oublier qu'en général, plus d'un

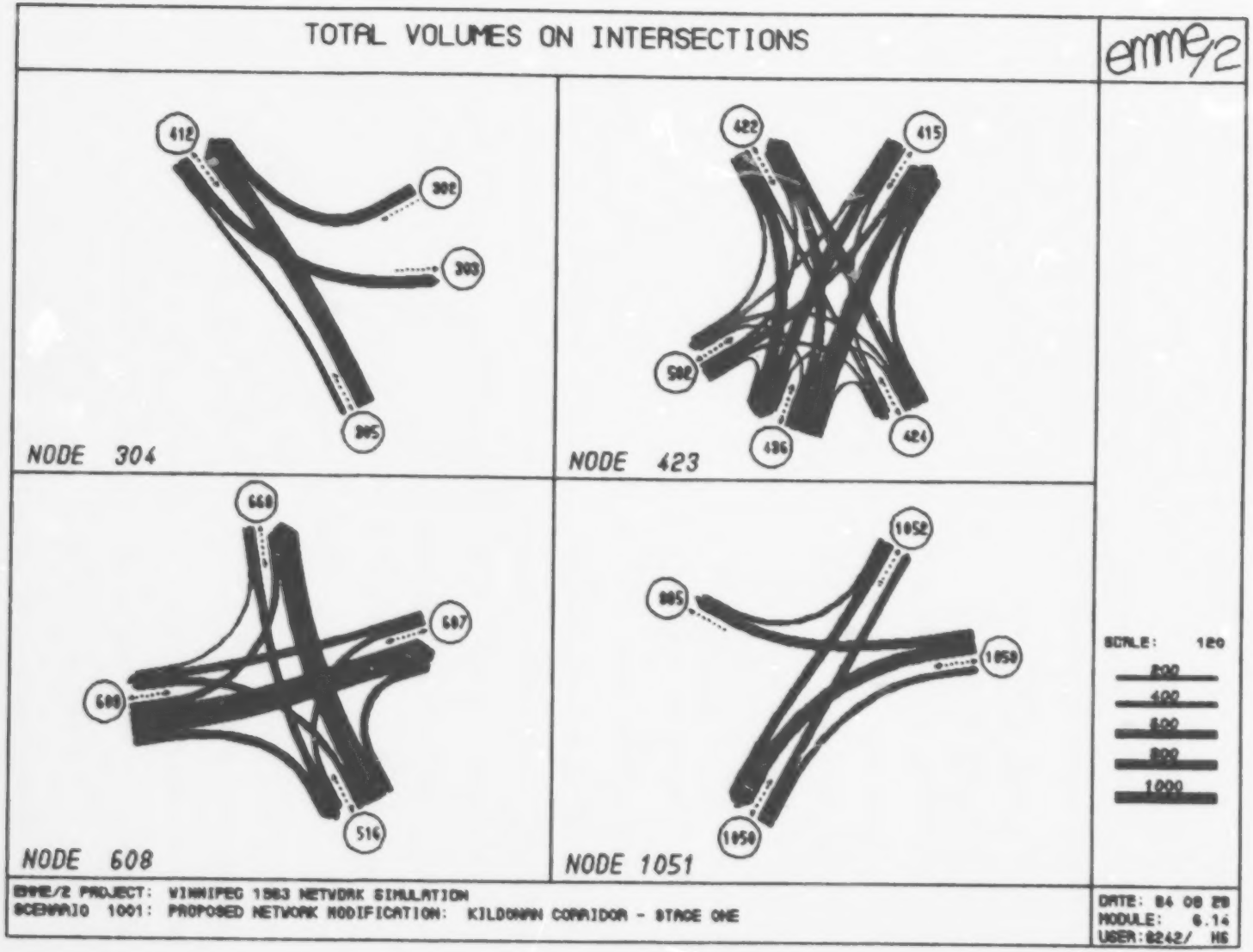

FIGURE III-5-16: Volumes sur les virages des carrefours.

plus court chemin existe - fait découlant du principe de Wardrop - et que le dessin n'en montre qu'un seul, choisi arbitrairement. Une deuxième façon d'utiliser ce module consiste dans l'analyse d'accessibilité. Il s'agit ici d'identifier les centroïdes qui, à partir d'un centroïde racine, peuvent être atteints en un certain temps (ou distance) donné. Pour faire cela, l'usager entre un seuil de temps (ou distance) et seulement les chemins vers les centroïdes qui peuvent être atteints sans dépasser le seuil sont dessinés. Par la suite, l'usager peut utiliser la commande graphique "C" (voir section 4.4) pour, pas à pas, y ajouter les centroïdes qui peuvent être atteints en bas de deux fois la valeur seuil, trois fois ,... etc. Sur un écran monochrome, il faut garder des copies de chaque tranche pour pouvoir les comparer par la suite. Sur un écran polychrome, chaque tranche du dessin est affichée en une couleur différente, permettant ainsi de distinguer les classes d'accessibilité sur le dessin final. Puisque ce module tient compte des pénalités de virages, on ne considère que les chemins entre centroïdes, qui, par définition, n'ont pas de virages pénalisés. Ceci est exploité en plus, graphiquement, par le fait qu'on n'affiche que les boîtes de noeuds des centroïdes; les noeuds réguliers ne servant que de points de support pour l'affichage des points. Pour rendre plus lisibles les détails des chemins aux noeuds, les virages des chemins sont affichés par des bouts de courbes splines, reliant les deux liens adjacents. Cette méthode innovatrice d'affichage de chemins permet, a), . de visualiser exactement les chemins calculés (qui peuvent contenir des cycles à cause des pénalités de virage) et b), d'afficher l'information d'une façon plus dense qu'il ne serait possible en affichant les noeuds réguliers. Deux exemples servent comme illustration de ce module. Dans la figure III-5-17, on reconnaît les plus courts chemins du centre-ville vers tous les

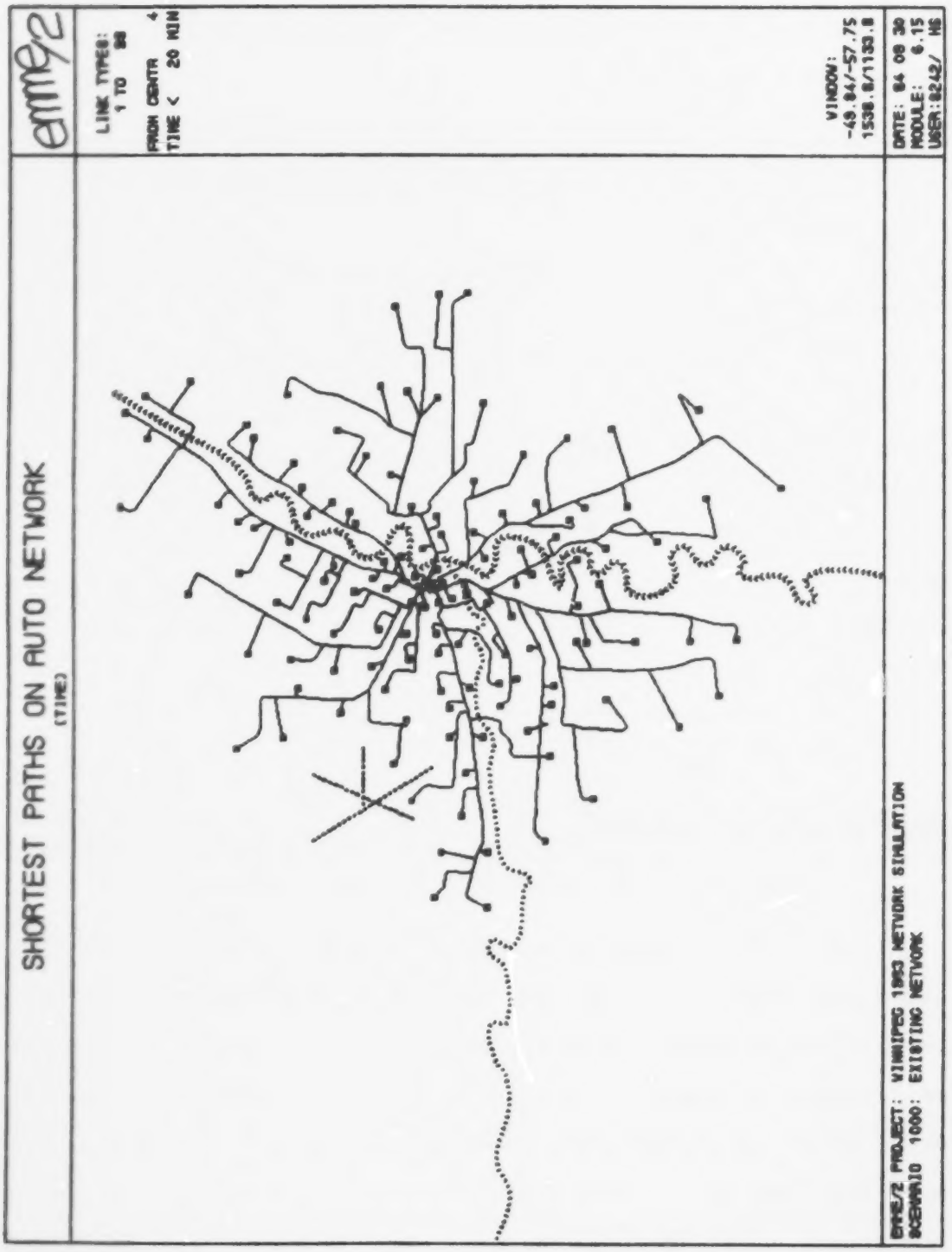

FIGURE III-5-17: Plus courts chemins et accessibilité des centroïdes.

centroïdes qui peuvent être atteints en moins de 20 minutes. La figure III-5-18 nous montre des détails des chemins, qui peuvent prendre des formes assez compliquées.

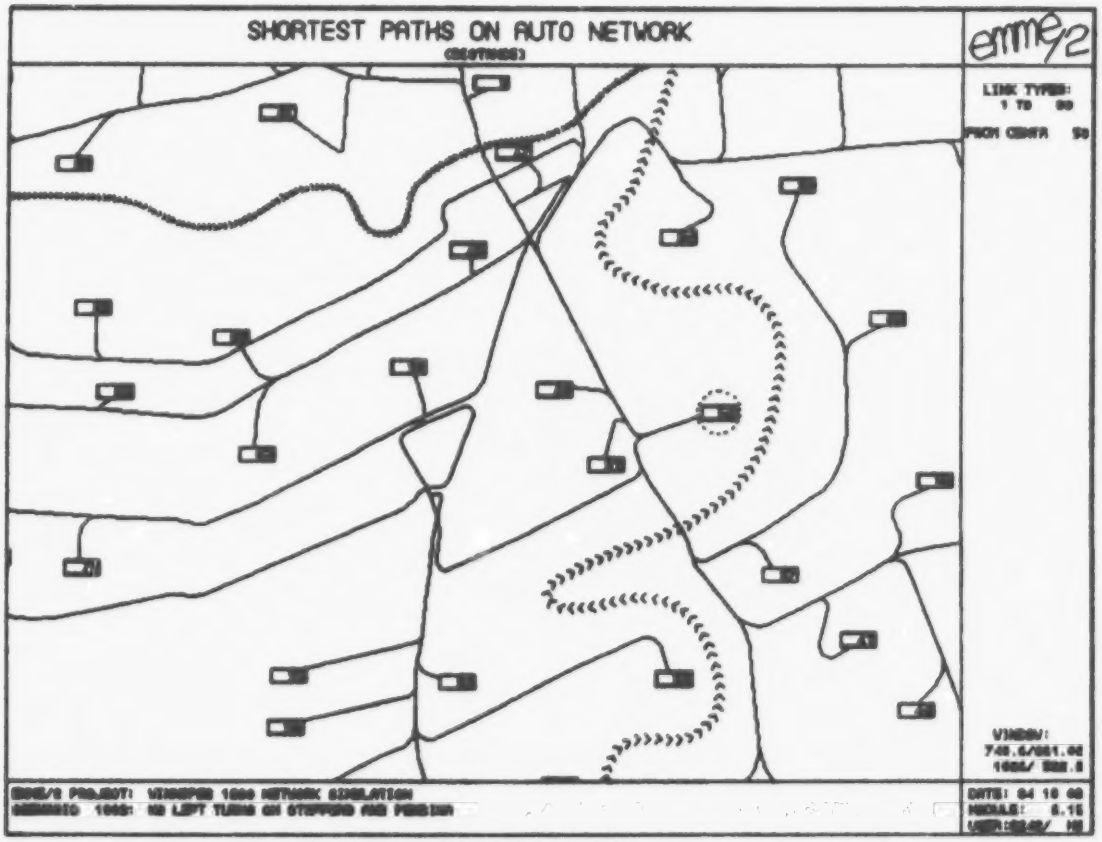

FIGURE III-5-18: Détails des chemins du mode auto.

Le module 6.21, "Rapports de temps et volumes de transport en commun", génère des rapports concernant les résultats d'une affectation des voyages par transport en commun. On distingue deux types de volumes: les volumes de transport en commun proprement dits, i.e. les volumes sur les segments de lignes, et les volumes auxiliaires qui proviennent des déplacements entre les lignes, en utilisant les modes auxiliaires de transport en commun (par exemple, accès, egrès, marche à pied, etc.). Quatre types de rapport sont disponibles:

- résumé de l'utilisation des lignes de transport en commun,
- temps et volumes au long des itinéraires des lignes,
- volumes sur le réseau de base,
- activités sur les noeuds.

Le module 6.22, "Affichage graphique des volumes de transport en commun", permet à l'usager de visualiser l'utilisation du réseau de transport en commun, telle que prédite par le modèle d'affectation. L'affichage des volumes est fait au niveau des liens et au niveau des noeuds. Sur les liens, l'usager a le choix entre

- les volumes de transport en commun d'un sous-ensemble de lignes,
- les volumes auxiliaires provenant d'un sous-ensemble de modes auxiliaires de transport en commun,
- les volumes totaux sur le réseau de base, soit la somme des volumes de toutes les lignes et des volumes auxiliaires.

Les volumes sur les noeuds représentent les activités suivantes:

- nombre d'embarquements initiaux,
- nombre d'embarquements pour but de transfert,
- nombre total d'embarquements,
- nombre de passager restant dans les véhicules,
- nombre de débarquements finaux,
- nombre de débarquements pour but de transfert,
- nombre total de débarquements.

Le premier exemple, voir figure III-5-19, montre les volumes sur une seule ligne de transport en commun. Le deuxième exemple, voir figure III-5-20, contient, pour chaque noeud, le nombre de passagers montant à bord pour la première fois au long de leur voyage.

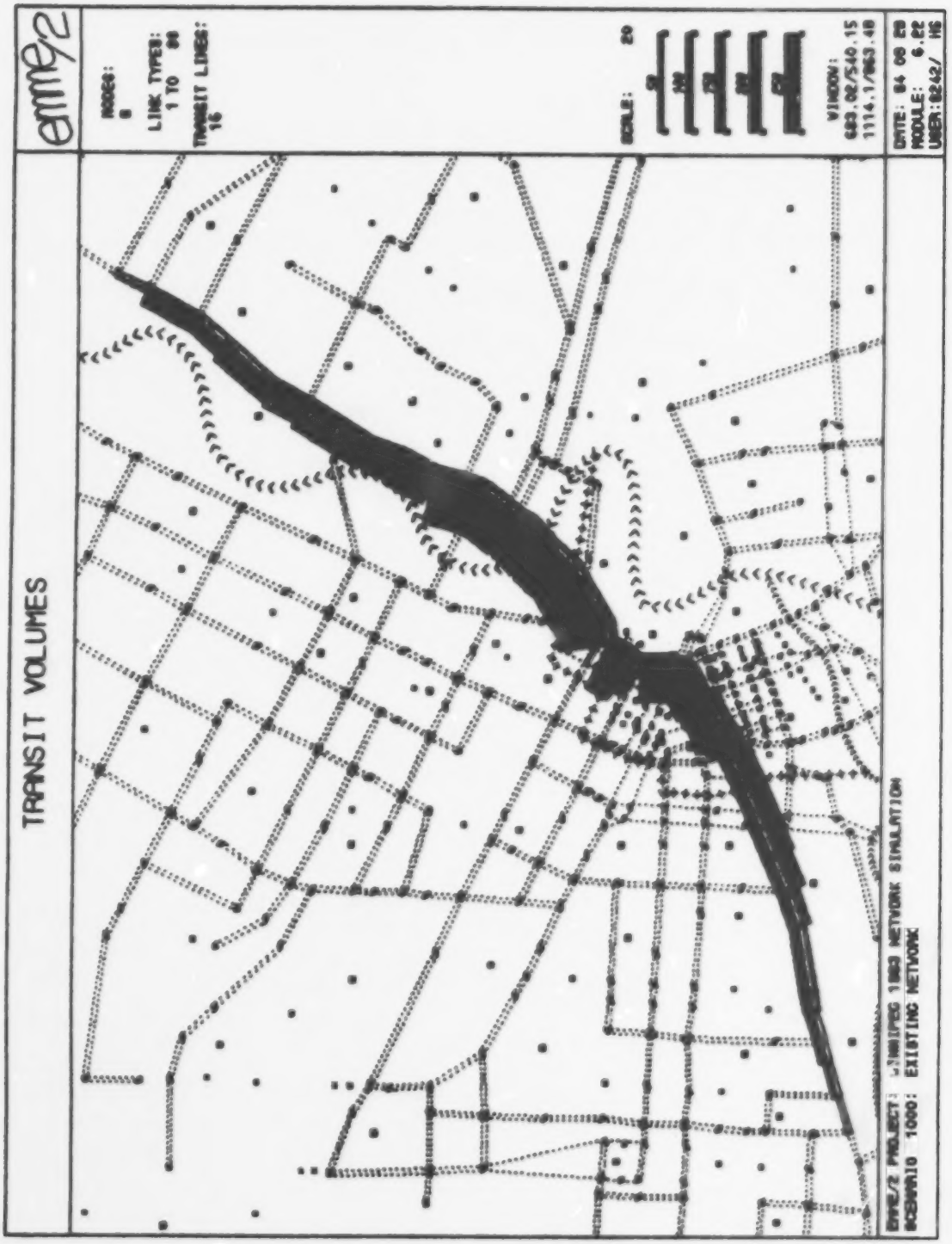

FIGURE III-5-19: Les volumes sur une ligne de transport en commun.

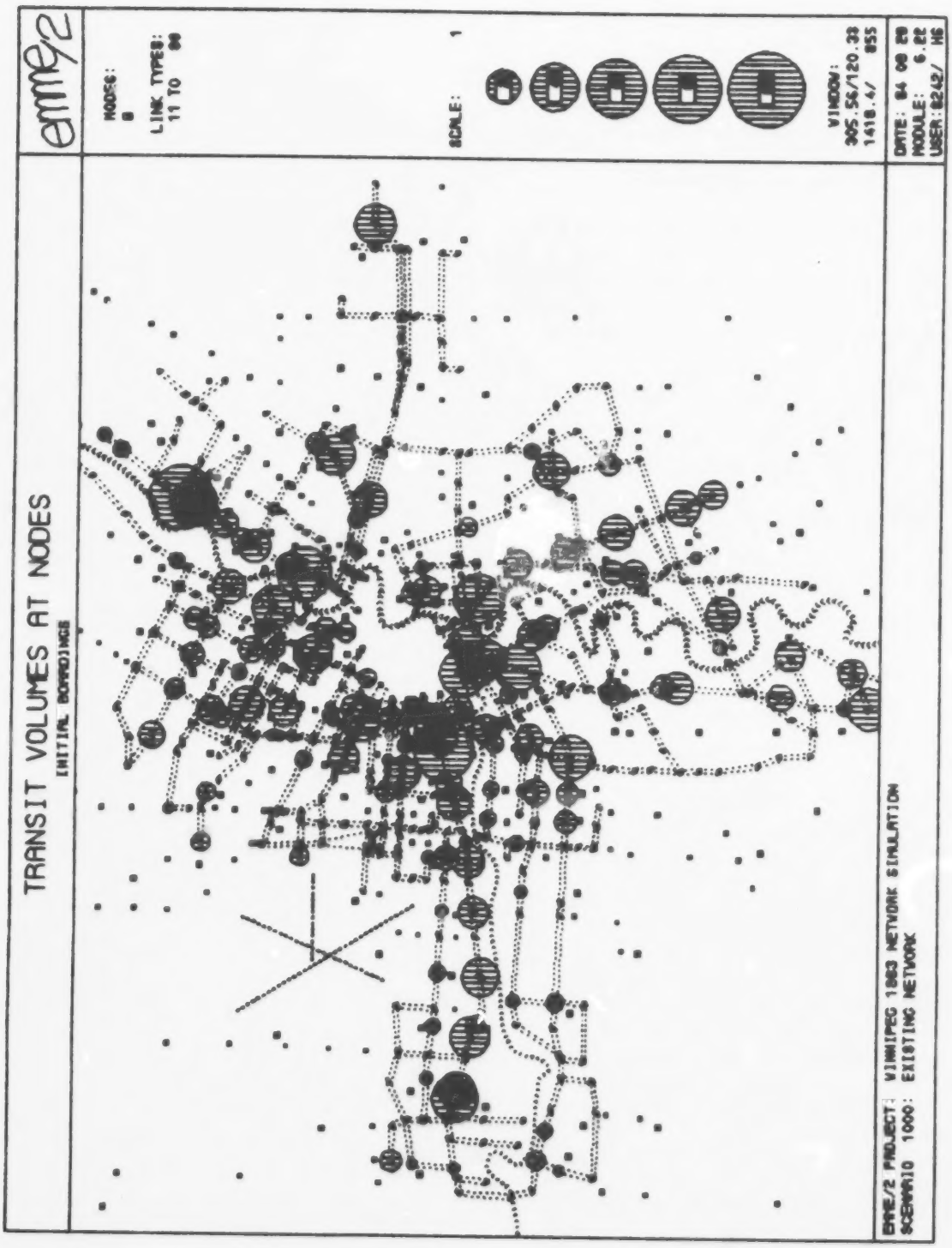

FIGURE III-5-20: Nombre d'embarquements initiaux.

6. Les applications

Même si les travaux de développement du système EMME/2 sont encore en cours, le système a déjà trouvé plusieurs applications, soit dans le cadre de projets de recherche au CRT, soit dans des agences de planification de transport. Dans cette section, nous présenterons brièvement les divers utilisateurs que le système EMME/2 a trouvé à date, et nous décrirons leur contexte de planification ainsi que l'environnement informatique dans lequel fonctionne le logiciel.

Au moment de la rédaction de ce texte (octobre 1984), le système EMME/2 est installé à quatre endroits hors de l'Université de Montréal, où il sert à des agences de planification de transport des villes suivantes:

- Montréal
- Vancouver
- Stockholm (Suède)
- Portland (Orégon)

En mai 1982, le *Conseil des Transports de la Région de Montréal* (COTREM) a décidé d'implanter le logiciel EMME/2 comme outil pour ses activités de planification. Pour permettre le traitement efficace du réseau complet de la région de Montréal, qui dessert une population d'environs 2.5 Mio représentée par 700 zones, le logiciel fut installé sur un ordinateur de grande puissance, soit le système IBM 3083 de la *Régie d'Assurance Automobile du Québec* (RAAQ) à Québec. Un terminal graphique monochrome de très haute résolution, de type Tektronix 4116, se trouve dans les bureaux du COTREM et communique par ligne téléphonique dédiée avec l'ordinateur central à Québec. Le système est utilisé pour des études

depuis décembre 1983. Entre temps, l'organisme a changé de nom: il s'appelle maintenant *Direction Générale du Transport Terrestre des Personnes* (DGTTP). En septembre 1984, à l'occasion du Congrès International des Transports, la DGTTP, en collaboration avec le Ministère du Transport du Québec, présenta le système EMME/2 au public (Primeau et al., 1984). La figure III-6-1 montre la distribution spatiale des embarquements initiaux pour le transport en commun sur l'île de Montréal.

Aussi au printemps 1982, une première version du logiciel EMME/2 fut installée à Stockholm, pour aider à planifier les réseaux de transports dans la région de la capitale suédoise. L'organisme responsable pour l'utilisation de EMME/2 est l'*Office du Trafic du Conseil du Comté de Stockholm* (*Trafikkontoret Stockholms Lans Landsting*). L'installation originale a été faite utilisant un ordinateur CDC CYBER 720 et une station graphique monochrome de type Tektronix 4154. Entre temps, l'organisme s'est équipé d'un mini-ordinateur VAX 11/750 et d'un terminal couleur Tektronix 4115B, qu'il compte utiliser en même temps pour le comté de Stockholm et pour plusieurs applications de contexte régional et national. La figure III-6-2 montre les volumes prédits sur les lignes de métro dans la ville de Stockholm.

En été 1983, le *Metropolitan Service District* de la ville de Portland en Orégon s'est doté d'un micro-ordinateur PIXEL 100/AP dans le but d'utiliser le système EMME/2 comme leur principal outil de planification.

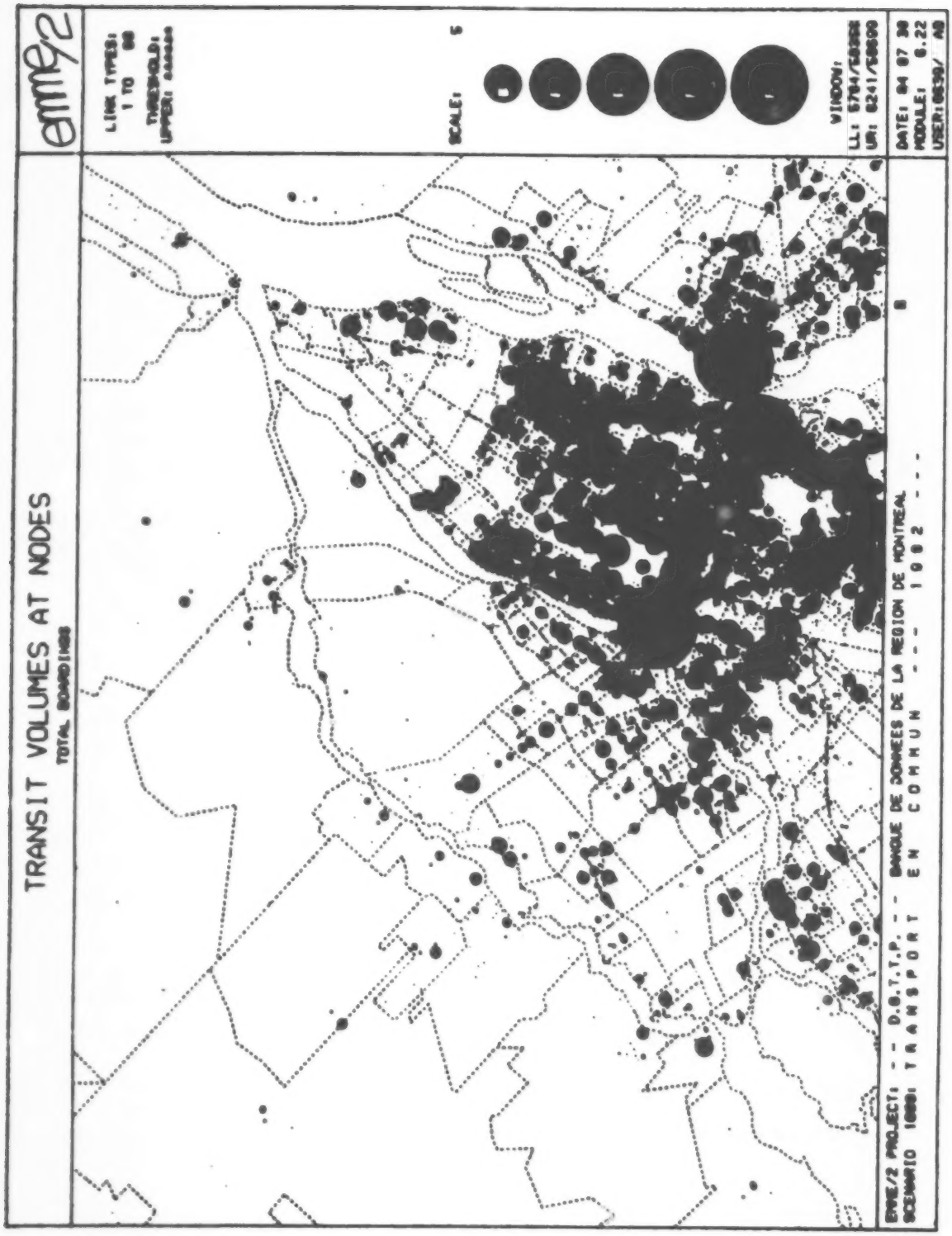

FIGURE III-6-1: Nombre d'embarquements sur le réseau t.c. à Montréal.

FIGURE III-6-2: Volumes sur les lignes de métro à Stockholm (Suède).

La région, dont cet organisme est responsable, comprend une population de 1.2 Mio et est modélisée par un ensemble de 390 zones origines et destinations. Le système est aussi utilisé par la compagnie de transport public de la ville de Portland et par le département de transport de l'état d'Orégon. La figure III-6-3 donne une idée de la taille du réseau considéré.

Depuis le printemps 1984, le système EMME/2 est implanté à Vancouver B.C., pour servir dans le cadre des études du *Greater Vancouver Regional District*, l'organisme responsable pour la planification des transports de la ville de Vancouver et ses banlieues. La région, qui comprend environ 1.4 Mio d'habitants, est sous-divisée en quelque 350 zones. Le logiciel est, comme dans la cas de Portland, installé sur un micro-ordinateur de type PIXEL 100/AP. Des terminaux graphiques de type Tektronix 4012, Tektronix 4014, et Visual 550 y sont reliés. Une fenêtre du réseau de Vancouver est montré dans la figure III-6-4.

A l'Université de Montréal, le système EMME/2 est installé sur deux ordinateurs: Le CDC Cyber 173 du centre de calcul et le PIXEL 100/AP du CRT, où il a été - et l'est toujours - utilisé dans le cadre de plusieurs projets de recherche.

Dès le début du développement de EMME/2, les données de la ville de Winnipeg nous ont servi comme principal cas de test. A part son rôle de "cobaye", la banque de données de Winnipeg fut aussi utilisée, à plusieurs occasions, par les planificateurs de transport de la ville de Winnipeg pour analyser des changements de réseau proposés.

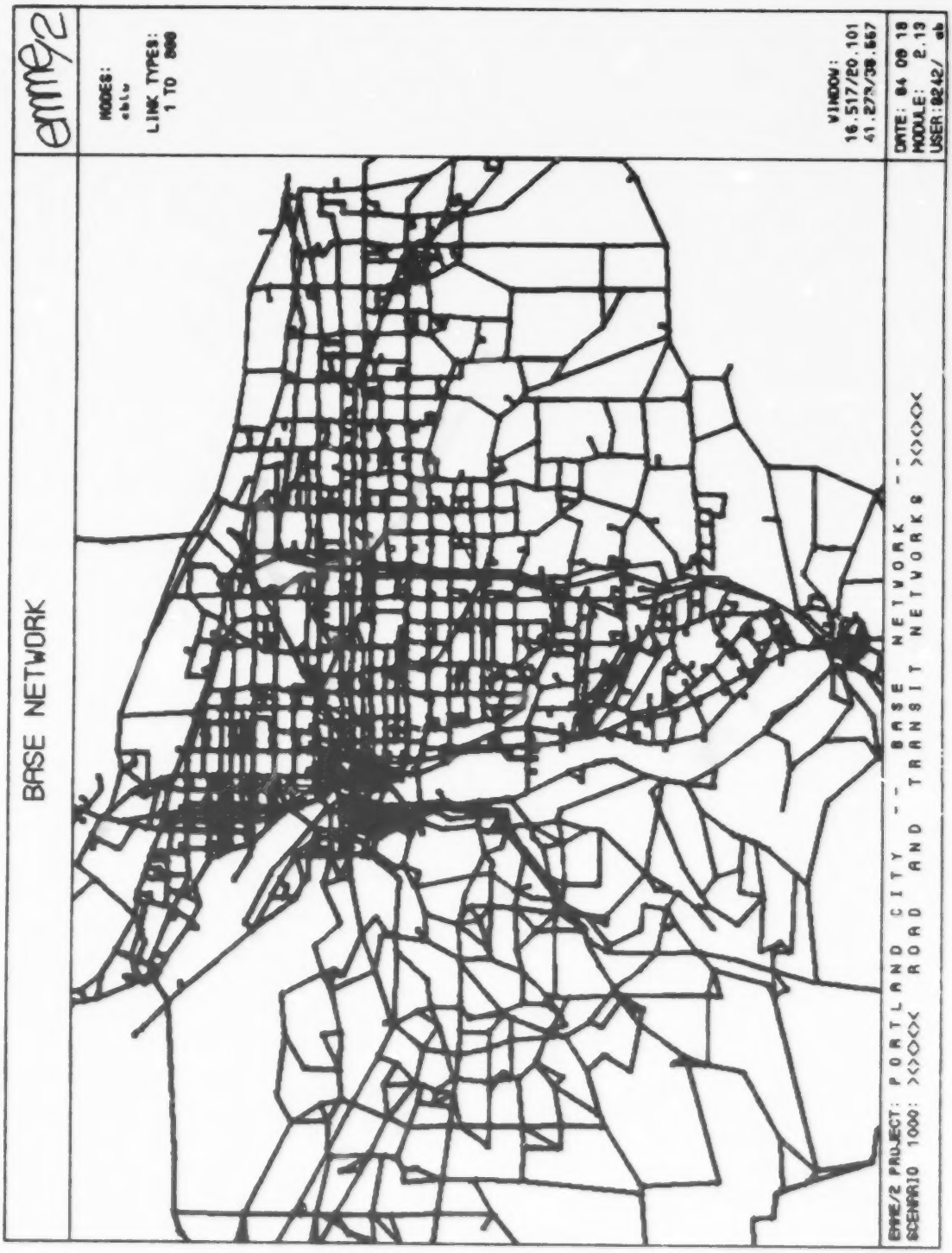

FIGURE III-6-3: Le réseau de base de la ville de Portland (Orégon).

FIGURE III-6-4: Une fenêtre du réseau de base de la ville de Vancouver.

En hiver 1982/1983 le CRT collabora à une étude de rationalisation des transports publics de la ville de Trujillo (Pérou). Cette étude, financée par C.I.D.A. (Canadian International Development Agency), avait pour but de réorganiser complètement le système de transport public de Trujillo. Les lignes de transport en commun existantes, chacune opérée par une compagnie différente, suivaient des itinéraires longs et indirects, à fin de maximiser la partie de la ville desservie par chaque compagnie. La figure III-6-5 montre l'itinéraire d'une telle ligne, ainsi que les volumes sur ses segments. L'étude de rationalisation (Guemili, 1983) concluait qu'en réorganisant le réseau et en utilisant des véhicules modernes, il était possible de diminuer la longueur totale des itinéraires de 28% et le nombre de véhicules de 33%, tout en diminuant le temps de voyage moyen de 30%. Pour ce projet, le système EMME/2 n'a été utilisé qu'avec le réseau de transport en commun, en utilisant des vitesses de parcours constantes.

Depuis juin 1982, le CRT collabore avec la *Régie Autonome des Transports Parisiens*, (RATP), sur une étude des conséquences de diverses stratégies de régulation du stationnement des voitures sur le terrain public. A ce but, une banque de données a été mise sur pied qui contient, dans une première étape, le réseau routier de la ville de Paris et ses banlieues. La figure III-6-6 montre les volumes de trafic routier au centre-ville, pendant l'heure de pointe du soir.

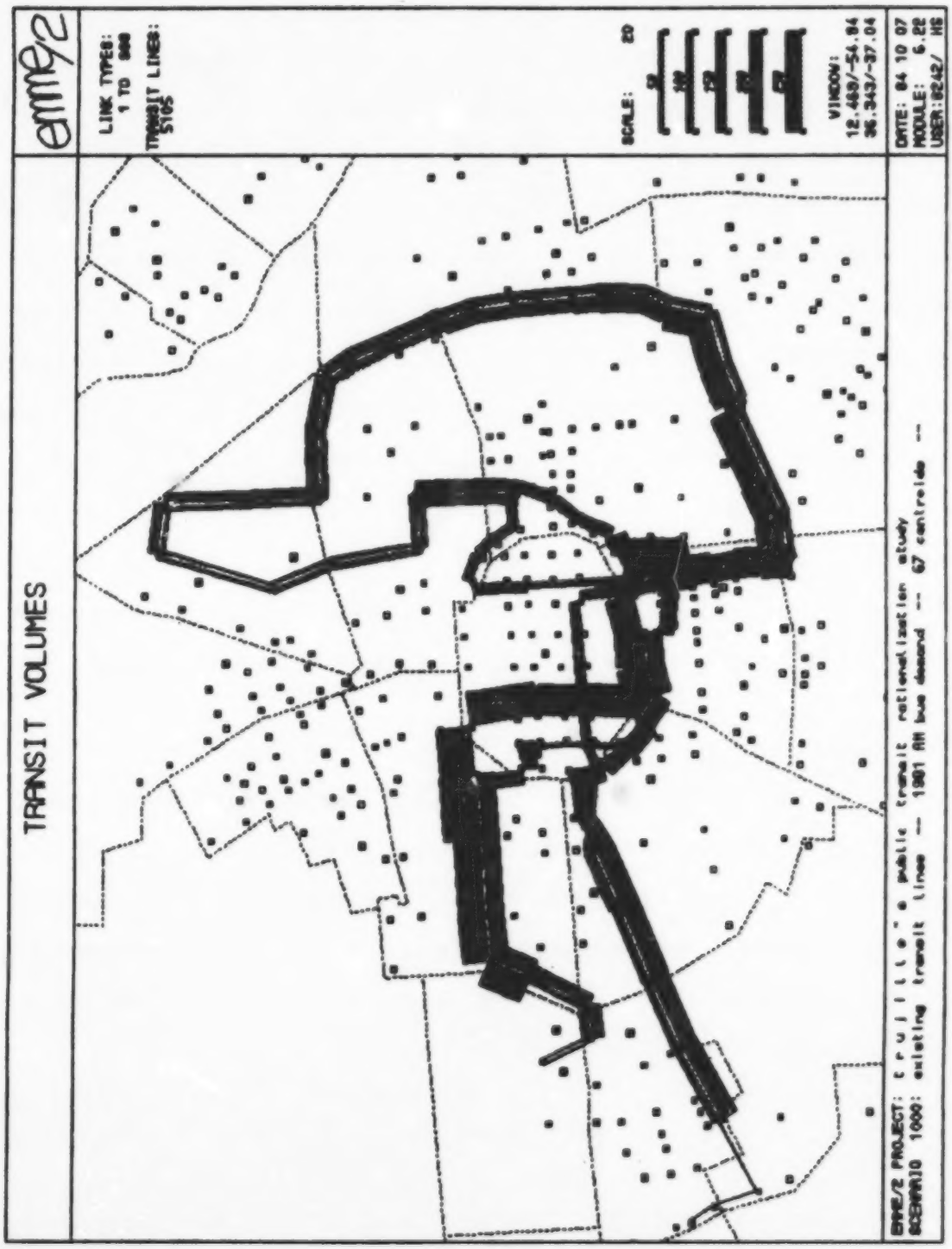

FIGURE III-6-3: Itinéraire et volumes d'une ligne d'autobus à Trujillo (Pérou) avant l'étude de rationalisation.

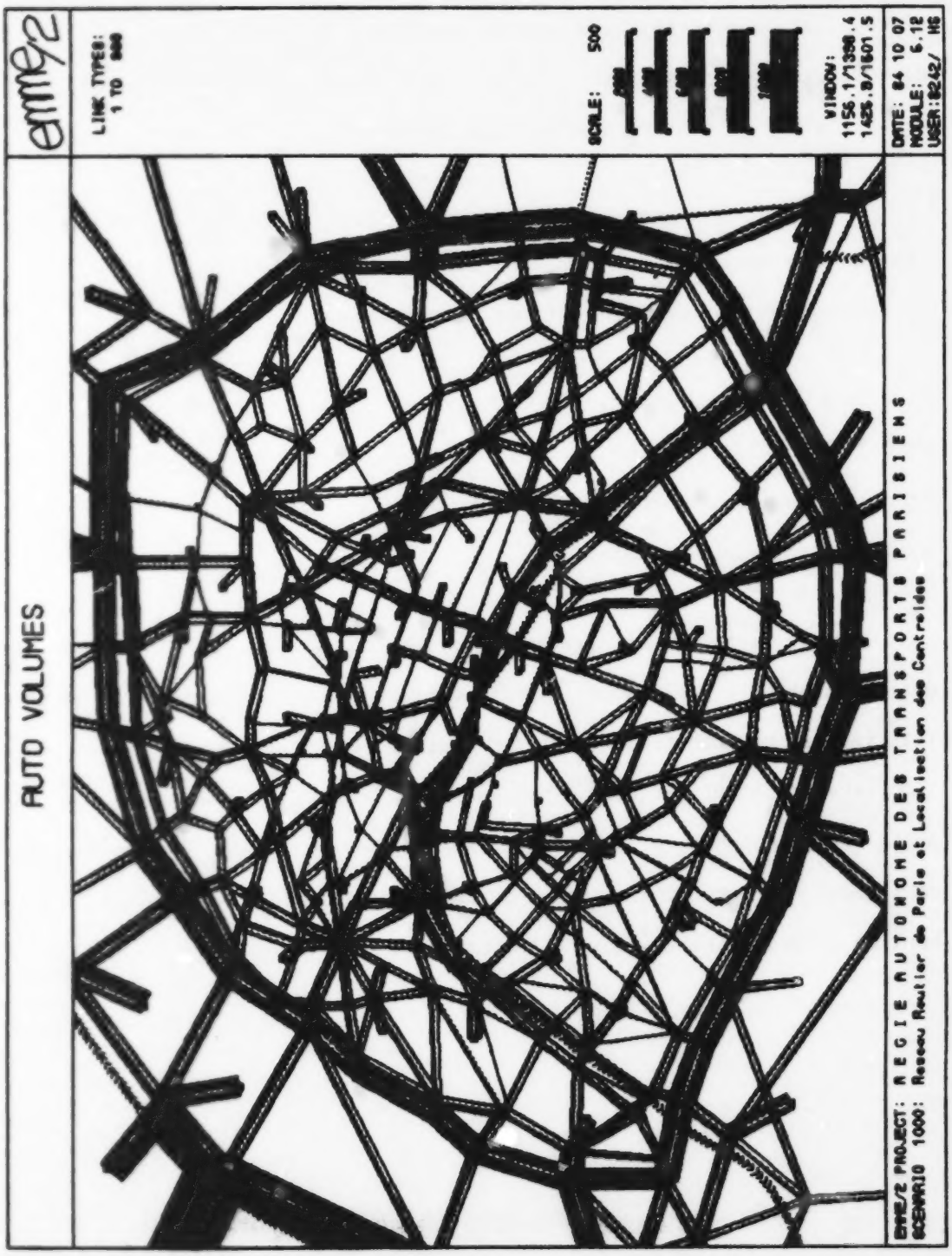

FIGURE III-6-6: Volumes - auto dans la région métropolitaine de Paris.

Grâce à leurs capacités interactives - graphiques pour l'édition de réseaux et de matrices, les modules de EMME/2 ont aussi trouvé application dans le cadre de projets qui, en soi, ne sont pas reliés directement à la planification du transport urbain des personnes.

L'éditeur de réseau de EMME/2 a servi pour la mise au point d'un réseau routier de la ville de Montréal qui, par la suite, fut intégré dans un système gestion du transport des handicapés.

Dans le cadre du projet FRETNET (Hamelin et al., 1981), le système EMME/2 fut utilisé pour la création et la gestion d'un réseau multimodal pour représenter les mouvements des marchandises par camion, voie ferrée et avion à l'intérieur du Canada.

Dans le cadre d'un projet de recherche toujours en cours, EMME/2 est utilisé intensivement pour le traitement de réseaux et matrices. Ce projet, fait en collaboration avec le Ministère du Transport du Brésil et l'Université Catholique de Rio de Janeiro, a pour but d'analyser les mouvements de marchandises au Brésil et, dans une étape ultérieure, de développer un outil interactif - graphique pour la planification du transport de marchandises, basé sur l'approche de EMME/2.

7. Conclusions

Dans ce chapitre, nous avons présenté EMME/2, un système interactif-graphique pour la planification de transport urbain de personnes. Il s'agit d'un ensemble d'outils intégrés, qui permettent au planificateur de définir, de valider et d'évaluer des scénarios de réseaux de transport multimodal. Contrairement aux systèmes de traitement par lots qui, en général, se limitent à l'implantation d'un modèle d'affectation, le but principal de EMME/2 est de créer un *environnement de travail*, à l'intérieur duquel peuvent être effectuées toutes les étapes d'un exercice de planification.

Nous avons montré comment, à l'aide d'un petit ensemble de concepts de base très simples, il est possible d'implanter un tel système. La simplicité des concepts nous a amené à un logiciel hautement portable, avec des points d'interface précisément définis. Nous avons étudié les besoins particuliers d'un mode de travail interactif-graphique et nous avons vu qu'il en découle des conséquences profondes sur la structure du logiciel, des entrées / sorties et des données. De plus, puisque le système interagit directement avec le planificateur, il est essentiel de bien structurer les types d'interaction entre les deux. Il s'agit de donner une "personnalité" au système, qui soit simple et cohérente, pour que l'usager se sente à l'aise dans son nouvel environnement de travail. Dans ce but, nous avons formulé un petit ensemble de règles d'interaction simples, qui ont été appliquées uniformément et strictement à travers tout le système.

Pour permettre un traitement graphique efficace de données de réseau de transport, nous avons développé un ensemble de primitives graphiques,

appelé OPLOT, qui est adapté aux besoins particuliers de ce type de dessin.

Le système EMME/2 a été implanté sur des ordinateurs de grande et moyenne taille (CDC Cyber, IBM, VAX) ainsi que sur des micro-ordinateurs de haut de gamme (PIXEL). Puisque l'implantation sur micro-ordinateur permet de traiter aisément des réseaux de taille considérable (400 zones, 2500 noeuds, 8000 liens, 300 lignes de transport en commun avec un total de 15000 arrêts), nous avons démontré qu'un tel système peut être accessible, à coût d'équipement modeste, aux agences de planification qui, antérieurement, n'ont pas eu accès à des installations informatiques.

L'installation du système EMME/2 dans les agences de planification de transport des villes de Montréal, Vancouver, Stockholm et Portland (Orégon) s'est démontrée très fructueuse. Les planificateurs utilisent le système EMME/2 pour définir, valider, affecter et analyser des scénarios de réseaux. Grâce au mode de travail interactif-graphique, le nombre de scénarios qu'ils peuvent analyser dans un temps raisonnable a augmenté considérablement, permettant ainsi des études plus approfondies. De plus, le système EMME/2 a servi comme outil important dans le cadre de plusieurs projets de recherches au Centre de recherche sur les transports.

Bien que le développement du système EMME/2 est à un stade très avancé, il reste à faire l'ajout de certains modules pour compléter le système.

Dans l'éditeur de réseau, un module de *calculatrice interactive de réseau* a été conçu, mais n'est pas encore programmé. A l'aide d'une approche similaire à celle de la calculatrice interactive de matrices

(module 3.21), ce module permettra à l'usager d'effectuer des modifications systématiques sur les attributs du réseau. L'usager pourra entrer des expressions algébriques et logiques pour, soit calculer des résultats immédiats, soit créer ou modifier des ensembles de données dans la banque de données. Des règles de conversion entre les divers niveaux (mode, noeud, lien, véhicule, ligne, arrêt) premettront l'agrégation et l'expansion automatique.

Dans l'éditeur de matrices, on envisage d'ajouter un modèle d'estimation de matrices O-D, à partir de volumes observés sur le réseau. Ce module sera basé sur le modèle développé dans la section 2 du chapitre II.

Bien que toute l'infrastructure de données nécessaires soit déjà en place, le module d'affectation d'équilibre multimodal n'est pas encore programmé. La première étape pour y arriver, sera d'adapter les algorithmes d'affectation pour qu'ils puissent traiter le cas d'une demande variable. Du côté transport en commun, nous avons analysé, dans la section 1.6 du chapitre II, une extension non-linéaire pour le modèle d'affectation. Cette extension du modèle pourrait facilement être intégrée dans EMME/2 pour tenir compte de certains effets de congestion sur le réseau de transport en commun. Du côté transport routier, il serait avantageux de développer une méthode alternative pour la définition des virages pénalisés, qui permettrait d'interdire systématiquement des virages à gauche sur un corridor, sans nécessairement considérer tous ces noeuds comme intersections.

Pour compléter les possibilités d'analyse graphique des résultats

d'une affectation, nous ajouterons d'ici peu un module qui permet la comparaison des volumes de transport en commun entre deux scénarios, ainsi qu'un module qui permet de visualiser les profils de charge des lignes individuelles de transport en commun.

BIBLIOGRAPHIE

AASHTIANI, H.Z. et MAGNANTI, T.L., *Equilibria on a Congested Transportation Network*, SIAM J. Algebraic Discrete Math 2, 213-216, 1981.

AHO A.V. et ULLMAN J.D., **Principles of Compiler Design**, Addison-Wesley, Reading, Mass. 1977.

ANDREASSON I., *A Method for the Analysis of Transit Networks*, 2nd European Congress on Operations Research (Mark Roubens, ed.), North Holland, 1976.

ANDREASSON I., *The Volvo Approach to Computer-Aided Transit Planning*, Transportation Research Record 657, 1978.

BABIN A., FLORIAN M., JAMES-LEFEBVRE L. et SPIESS H., *EMME/2: Interactive Graphic Method for Road and Transit Planning*, Transportation Research Record 866 pp 1-9, 1982.

BABIN A., FLORIAN M., JAMES-LEFEBVRE L. et SPIESS H., *EMME/2: User's Manual*, Centre de recherche sur les transports, Université de Montréal, 500p, 1982 (and later updates).

BREGMAN L., *The Relaxation Method of Finding the Common Point of Convex Sets and its Application to the Solution of Problems in Convex Programming*, U.S.S.R. Computational Math. Mathematical Phys. 7, pp 200-217, 1967.

BUREAU OF PUBLIC ROADS, *Traffic Assignment Manual*, U.S. Department of Commerce, Washington D.C., 1964.

CHAPLEAU R., *Réseaux de Transport en Commun: Structure Informatique et Affectation*, thèse de doctorat, Dép. I.R.O., Université de Montréal, 1974.

CHRIQUI C., *Réseau de Transport en Commun: Les Problèmes de Cheminement et d'Accès*, thèse de doctorat, Dép. I.R.O., Université de Montréal, 1974.

CHRIQUI C. et ROBILLARD P., *Common Bus Lines*, Transportation Science 9, pp 115-121, 1975.

CRVCANIN M., *IBR - Interactive Graphic Routines*, ITEP, Ecole Polytechnique Fédérale, Lausanne, 1978.

DAFERMOS, S.C., *Traffic Equilibrium and Variational Inequalities*, Transportation Science 14, pp 42-54, 1980.

DEMING W.E. et STEPHAN F.F., *On a Least Square Adjustment of a Sampled Frequency Table when Expected Marginal Totals are Known*, Annal of Mathematical Statistics, Vol 11, pp 427-444, 1940.

DIAL R.B., *Transit Pathfinder Algorithm*, Highway Research Record 205, pp 67-85, 1967.

DIAL R.B., *A Probabilistic Multipath Traffic Assignment Model which Obviates Path Enumeration*, Transportation Research Vol 5, 1971.

DIAL R.B., *NED: An Interactive Network Edit-and Display Program for the Apple II Personal Computer*, Transportation Research Record , 1982.

DIJKSTRA E.W., *A Note on Two Problems in Connection with Graphs*, Numerische Mathematik 1, pp 269-271, 1959.

DOUGLAS G.B., *Applications of Interactive Graphics in Urban Transit Analysis*, Transportation Research Record 866, pp18-25, 1982.

EVANS S.P. and KIRBY H.R., *A Three-Dimensional Furness Procedure for Calibrating Gravity Models*, Transportation Research 8, pp 105-122, 1974.

FEARNSIDE K. et DRAPER D.P., *Public Transport Assignment - A New Approach*, Traffic Engineering and Control, pp 298-299, November 1971.

FISK C. et NGUYEN S., *Solution Algorithms for Network Equilibrium Models with Asymetric User Costs*, Publication #167, Centre de recherche sur les transports, Université de Montréal, 1980.

FLORIAN M., *A Traffic Equilibrium Model of Traffic by Car and Public Transit Modes*, Transportation Research 10, pp 166-179, 1977.

FLORIAN M., CHAPLEAU R., NGUYEN S., ACHIM C., JAMES-LEFEBVRE L., GALARNEAU S., LEFEBVRE J. et FISK C., *Validation and Application of EMME: An Equilibrium Based Two-Mode Urban Transportation Planning Method*, Transportation Research Record 728, 1979.

FLORIAN M. et SPIESS H., *On Binary Mode Choice / Assignment Models*, Transportation Science 17, pp 32-47, 1983.

FLORIAN M. et SPIESS H., *The Convergence of Diagonalization Algorithms for Assymetric Network Equilibrium Problems*, Transportation Research B 16B, pp 477-483, 1982.

FRANK M. et WOLFE P., *An Algorithm for Quadratic Programming*, Naval Research Logistics Quarterly 3, pp 95-110, 1956.

FURNESS K.P., *Time Function Interaction*, Traffic Engineering and Control, Vol 7, No 7, pp 19-36, 1970.

General Motors Transportation System Center, *Interactive Graphic Transit Design System (IGTDS): User's Manual*, General Motors Technical Center, Warren, MI, 1978.

GOTTS T. et ESCH R., *Applications of Interactive Graphics in Michigan's Statewide Transportation Modeling System*, Transportation Research Record 787, 1980.

GRAYSON A., *Interactive UTPS: Implementation under a Timesharing Environment*, Transportation Research Record 835, 1982.

GUEMILI M.A., *Public Transport Rationalization Study - Metropolitan Trujillo (Peru)*, M.A. Guemili & Associates Ltd., Winnipeg, 1983.

HALL M.D., VAN VLIET D. et WILLUMSEN L.G., *Assignment Model for the Evaluation of Traffic Management Schemes*, Traffic Engineering and Control 21, pp 168-179, 1980.

HAMELIN P., LOS M. et NGUYEN S., *Infrastructure d'un Réseau de Transport Multimodal pour le Canada*, publication #209, Centre de Recherche sur les Transports, Université de Montréal, 1981.

HASSELSTROM D., *Public Transportation Planning - A Mathematical Programming Approach*, Ph.D. thesis, Department of Business Administration, University of Gothenburg, Sweden, 1981.

JORNSTEN K.O. et NGUYEN S., *On the Estimation of a Trip Matrix from Network Data*, Centre de Recherche sur les Transports, Université de Montréal, Publication 153, 1979.

KIRBY H.R., *Theoretical Requirements for Calibrating Gravity Models*, Transportation Research, Vol 8, pp 97-104, 1974.

KNUTH D.E., The Art of Computer Programming - Volume 3 / Sorting and Searching, Addison-Wesley, Reading, Mass. 1973.

KORNHAUSER A.L. and HESS J.L., *Interactive Graphic Sketch Planning Model for Urban Transportation*, Transportation Research Record 657, 1978.

LAMOND B. et STEWART N.F., *Bregman's Balancing Method*, Transportation Research B, Vol 15B, pp 239-248, 1981.

LAMPKIN W. et SAALMANS P.D., *The Design of Routes, Service Frequencies and Schedules for a Municipal Bus Undertaking: A Case Study*, Operations Research Quarterly, Vol 18, No 4, pp 375-397, 1967.

LAST A. et LEAK S.E., *Transept: a Bus Model*, Traffic Engineering and Control, pp 14-20, January 1976.

LE CLERCQ F., *A Public Transport Assignment Method*, Traffic Engineering and Control, pp 91-96, June 1972.

LEBLANC L.J., MORLOK E.K. et PIERSKALLA E.P., *An Efficient Approach to Solving th Road Equilibrium Traffic Assignment*, Transportation Research 9, 309-318 1975.

LUENBERGER D.G., Introduction to Linear and Nonlinear Programming, Addison-Wesley, 1973.

MAGNENAT-THALMANN N. et THALMANN D., *A Graphical PASCAL Extension Based on Graphical Types*, Software - Practice and Experience 11, pp 53-62, 1981.

MANDL C.E., *Evaluation and Optimization of Urban Public Transportation Networks*, presented at the 3rd European Congress on Operations Research, Amsterdam, Netherlands, 1979.

MARCOTTE P., *Network Optimization with Continuous Control Parameters*, Transportation Science 17, pp 181-197, 1983.

MURCHLAND J., *The Multi-proportional Problem*, University College London, research note JDM 263, 1977.

NGUYEN S., *An Algorithm for the Traffic Assignment Problem*, Transportation Science 8, pp 203-216, 1974.

NGUYEN S., *Estimating Origin-Destination Matrices from Observed Flows*, Proceedings of the First Course on Transportation Planning Models of the International School of Transportation Planning, Amalfi, Italy, October 1982.

NGUYEN S. et JAMES L., *TRAFFIC - An Equilibrium Assignment Program*, Publication #17, Centre de recherche sur les transports, Université de Montréal, 1975.

POTTS B. and OLIVER M., *Flows in Transportation Networks*, Academic Press, New York, 1972.

RAPP M.H., *Transit System Planning: A Man - Computer Interactive Approach*, Highway Research Record 415, pp 95-103, 1972.

RAPP M.H., MATTENBERGER P., PIGUET S. et ROBERT-GRANDPIERRE A., *Interactive Graphic System for Transit Route Optimization*, Transportation Research Record 619, 1976.

RIEPL R.J. and GEHNER C.D., *Interactive Graphics in Europe: Three Computer-Aided Transportation Tools*, **Computer Graphics**, Vol. 1, No. 13, pp 18-25, 1978.

ROETHLISBERGER H., *Modèle de systèmes graphiques interactifs et application à la conception d'une station de mise en pages de textes composités*, Thèse de doctorat, Ecole Polytechnique Fédérale de Lausanne, Suisse, 1980.

SCHEELE C.E., *A Mathematical Programming Algorithm for Optimal Bus Frequencies*, Institute of Technology, University of Linköping, Sweden, 1977.

SCHNEIDER J.B., *Applications of Computer Graphics in the Transportation Field*, **Computer Graphics**, Vol. 1, No. 13, pp 18-25, 1978.

SCOTT J.E., **Introduction to Interactive Computer Graphics**, Wiley & Sons, New York, 1982.

SMITH D.P. et HUTCHISON B.G., *Goodness of Fit Statistic for Trip Distribution Models*, **Transportation Research A**, Vol 15A, pp 295-303, 1981.

SMITH, M.J., *The Existence, Uniqueness and Stability of Traffic Equilibria*, **Transportation Research B** 13B, pp 295-304, 1979.

TEKTRONIX Inc., *PLOT-10 Manuals*, Beaverton, Oregon.

TRIBOULT L., *DAVIS sous SITERT - Présentation*, Institut de Recherche des Transports, Département Informatique, Université de Grenoble, France, 1979.

UMTA/FHWA, *UTPS Reference Manual*, U.S. Department of Transportation, 1977.

VAN VLIET D. et WILLUMSEN L.G., *Validation of the ME2 Model for Estimating Trip Matrices from Traffic Counts*, Proceedings of The Eighth International Symposium on Transportation and Traffic Theory, June 1981.

VAN ZUYLEN H.J. et BRANSTON D.M., *Consistent Link Flow Estimation from Counts*, Transportation Research B, Vol 16B, pp 473-476, 1982.

VAN ZUYLEN H.J. et WILLUMSEN L.G, *The Most Likely Trip Matrix Estimated from Traffic Counts*, Transportation Research B, Vol 14B, pp 281-293, 1980.

VAN ZUYLEN H.J., *Some Improvements in the Estimation of an O-D Matrix from Traffic Counts*, Proceedings of The Eighth International Symposium on Transportation and Traffic Theory, June 1981.

WARDROP J.G., *Some Theoretical Aspects of Road Traffic Research*, Proc. Inst. Civil Engineers, Part II 1, pp 325-378, 1952.

WILKS S.S., Mathematical Statistics, Wiley & Sons, New York, 1962.

WILSON S.R., *Statistical Notes on the Evaluation of Calibrated Gravity Models*, Transportation Research, Vol 10, pp 343-345, 1976.

REMERCIEMENTS

Je tiens à remercier:

Mon directeur de recherche, Michael Florian. J'ai eu le privilège de profiter de son expérience immense, de son engagement extraordinaire pour ce projet et de jouir de sa confiance.

Linda James-Lefebvre, Isabelle Constantin et André Babin. Avec leur participation active et leur intérêt continu, ils ont contribué grandement au succès du projet EMME/2.

Sang Nguyen, Stefano Pallottino et Dirk Van Vliet. Leurs commentaires critiques et constructifs ont contribué au contenu théorique de cette thèse.

Le Centre de recherche sur les transports et le Département d'informatique et de recherche opérationnelle. En m'engageant comme agent de recherche et, pendant la dernière année, comme chargé d'enseignement, ils ont assuré mon support financier pendant ces études.

Le Conseil de recherches en sciences naturelles et en génie du Canada. Grâce à ses subventions d'appareillage, le système EMME/2 a pu être développé sur des équipements modernes.

Diane Larin, Denis Lebeuf et André Paquet. Ils m'ont aidé avec grande patience dans ma bataille continuelle avec la langue française.

Tous mes collègues du Centre de recherche sur les transports. Ils ont créé une ambiance de travail toujours agréable et stimulante.

Ma femme Barbara. Son encouragement et son support moral valent plus que je puisse l'exprimer dans une phrase.

Finalement, j'aimerais penser à mes parents et je me fais l'honneur de leur dédier ce travail.